Beiträge zu effizienten Algorithmen basierend auf Rang-1 Aufdatierungen

Von der Carl-Friedrich-Gauß-Fakultät
Technische Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte

D i s s e r t a t i o n

von

Dipl.-Math. Peter Stange,

geboren am 18.12.1979,

in Dresden

Eingereicht am:	30.06.2011
Mündliche Prüfung am:	04.10.2011
Referent:	Prof. Dr. Matthias Bollhöfer, TU Braunschweig
Korreferenten:	Prof. Dr. Daniel Kressner, École Polytechnique Fédérale de Lausanne
	Prof. Dr. Christian Mehl, TU Berlin

(2011)

Stange, Peter:
Beiträge zu effizienten Algorithmen basierend auf Rang-1 Aufdatierungen,
Dissertation,
Technische Universität Carolo Wilhelmina,
Braunschweig, 2011.

Zusammenfassung

Die Verwendung von Matrixfaktorisierungen, wie beispielsweise der Singulärwertzerlegung und der LU-Faktorisierung, spielt eine zentrale Rolle zur Lösung eines breiten Spektrums an Problemstellungen in der Numerik. Unter anderem sei hierzu die Lösung linearer Gleichungssysteme sowie die Berechnung des Rangs einer Matrix genannt. In der Regel ist ein relativ hoher Rechenaufwand erforderlich, um die genannten Faktorisierungen zu bestimmen. Speziell im Fall großer, dichtbesetzter Matrizen ergibt sich daraus selbst bei der Verwendung moderner Computersysteme ein recht hoher Zeitaufwand.

In verschiedenen Anwendungsgebieten ist es erforderlich zu einer Folge von Matrizen die zugehörigen Faktorisierungen zu berechnen, was zu einer weiteren Erhöhung der Rechenkomplexität führt. Unterscheiden sich diese Matrizen allerdings nur unwesentlich voneinander, so kann durch die Verwendung geeigneter Algorithmen der benötigte Rechenaufwand stark reduziert werden. Diese Verfahren beruhen darauf, dass die Zerlegung der modifizierten Matrix direkt aus der bekannten Faktorisierung der vorherigen Iteration berechnet werden kann.

Das zentrale Thema dieser Arbeit ist die Entwicklung effizienter Algorithmen zur Rang-1 Aufdatierung verschiedener Problemstellungen. Das heißt, den folgenden Aufgabenstellungen liegt jeweils eine Rang-1 Korrektur der Form

$$A_+ = A + ab^T$$

zugrunde. Die aufdatierte Matrix A_+ ergibt sich somit aus der ursprünglichen Matrix A sowie den Vektoren a und b, welche eine additive Rang-1 Korrektur ab^T bilden.

Im ersten Teil dieser Arbeit wird die Rang-1 Aufdatierung der LU-Faktorisierung betrachtet. Hierzu werden zunächst zwei bereits bekannte Algorithmen vorgestellt. Im Weiteren werden diese durch die Möglichkeit der Spaltenpermutation auf rechteckige Probleme erweitert sowie durch effiziente Matrixzugriffe in ihrer Laufzeit optimiert. Zusätzlich wird ein hybrider Algorithmus vorgestellt, welcher die Vorteile der verschiedenen Aufdatierungsverfahren vereint und speziell zur Verwendung in einem hier vorgestellten Optimierungspaket entwickelt wurde.

Im zweiten Teil der Arbeit wird die Rang-1 Aufdatierung der Singulärwertzerlegung für den Fall unsymmetrischer Matrizen betrachtet. Zur Entwicklung eines effizienten Aufdatierungsalgorithmus liegt hierbei das Hauptaugenmerk auf der schnellen Multiplikation von Matrizen. Dazu kann die Cauchy-Matrix Struktur der hier auftretenden Matrizen

ausgenutzt werden. Diese ermöglicht die Verwendung verschiedener Matrixapproximationen, welche im Hinblick auf die effiziente Multiplikation näher untersucht werden. Des Weiteren wird durch einen abschließenden Korrekturschritt eine hohe Lösungsgenauigkeit der aufdatierten Matrizen sichergestellt.

Im letzten Kapitel werden verschiedene Verfahren zur Rang-1 Aufdatierung der Matrixexponentialfunktion vorgestellt. Zunächst wird hierzu die Darstellung der Matrixfunktion durch das Cauchy-Integral verwendet. Es stellt sich heraus, dass die summierte Trapezregel ein geeignetes Verfahren zu dessen numerischer Berechnung ist. Der Schwerpunkt liegt in diesem Teil der Arbeit auf der theoretischen Untersuchung des Quadraturfehlers sowie der Darstellung effizienter Vorgehensweisen zur praktischen Verwendung dieser Aufdatierungsmethode. Dabei werden unter anderem die Wahl des Integrationswegs sowie die Lösung geshifteter Gleichungssysteme thematisiert. Abschließend wird als weiteres Aufdatierungsverfahren der Matrixexponentialfunktion eine auf dem Scaling und Squaring Algorithmus beruhende Methode vorgestellt. Diese ist nicht in jedem Fall anwendbar, ermöglicht aber unter gewissen Voraussetzungen eine sehr schnelle Aufdatierung.

Abstract

Matrix factorizations such as the LU decomposition or the Singular Value Decomposition are crucial for solving a broad range of numerical problems, among which are the computation of the rank of a matrix or solving linear systems. In general the determination of the factorizations are of high complexity. Especially, in case of large and dense matrices a lot of computational time is needed, even when using modern computer architectures.

In different applications factorizations for series of matrices have to be computed, which leads to an even higher complexity. However, if the matrices do not differ substantially from each other, efficient algorithms can be used to reduce the computational time. These algorithms are based on the idea of computing the decomposition of the modified matrix by directly using the factorization of the matrix from the previous iteration.

The main contribution of this work is the development of efficient algorithms for the rank-1 update in different applications. The underlying problem in each of these aplications is a rank-1 correction of the form

$$A_+ = A + ab^T.$$

The updated matrix A_+ is obtained from the original matrix A as well as two vectors a and b, which form an additive correction ab^T.

The first part of this work is concerned with the rank-1 update of the LU factorization. In the beginning two existing algorithms are presented. Then these methods are extended with the option of column pivoting, which is necessary for updating rectangular problems. The computational time is further reduced by efficient memory access. Additionally, a hybrid algorithm which combines the advantages of the different updating strategies is presented. It was developed for the use within an optimization code.

In the second part of this work the rank-1 update of the Singular Value Decomposition for unsymmetric matrices is investigated. The focus on the development of an efficient updating algorithm is put on the fast multiplication of matrices. To this end the Cauchy-matrix structure of some of the matrices can be exploited. This structure allows using different matrix approximations which are examined with respect to the efficient multiplication of matrices. To ensure the accuracy of the updated matrices an additional correction step is performed.

In the last chapter different methods for updating the matrix exponential are presen-

ted. At first the Cauchy-integral representation of the matrix function is used. It will be shown that the composite trapezoidal rule is an appropriate method for the numerical integration. The focus of this part of the work lies in the theoretical investigation of the quadrature error as well as in presenting efficient strategies for applying the updating method in practice. Regarding this, two points of interest are among others the choice of the path of the integration and solving shifted linear systems. In the end another updating algorithm, which is based on the Scaling and Squaring algorithm, is introduced. The approach can not always be applied, but under certain conditions it offers a very fast update of the matrix exponential.

Danksagung

Zunächst möchte ich mich bei meinem Doktorvater Matthias Bollhöfer bedanken, welcher mir jederzeit mit wertvollen Ratschlägen zur Seite stand und somit maßgeblich zum Gelingen meiner Arbeit beitrug. Mein weiterer Dank geht an Daniel Kressner, welcher mir das Thema der Aufdatierung von Matrixfunktionen näher brachte.
Ein großer Dank geht ebenfalls an alle meine Kollegen der AG Numerik, die mir in vielen fachlichen und weniger fachlichen Diskussionen Anregungen und Ideen lieferten sowie zu einer jederzeit angenehmen Arbeitsatmosphäre beitrugen.
Nicht zuletzt möchte ich mich bei meiner Familie, meiner Freundin sowie meinen Freunden bedanken. Sie haben mich nicht nur tatkräftig unterstützt, sondern auch für die erforderliche Abwechslung gesorgt.

Inhaltsverzeichnis

1 Einleitung **13**

1.1 Ziel der Arbeit 13

1.2 Struktur der Arbeit 14

2 Grundlagen **17**

2.1 Orthogonale Transformationen 17

2.1.1 Givens-Rotationen 17

2.1.2 Householder-Spiegelungen 18

2.1.3 Jacobi-Methode 19

2.2 LU-Faktorisierung 20

2.3 Singulärwertzerlegung 21

2.4 Sherman-Morrison-Woodbury Formel 26

2.5 Cauchy-Integral 26

2.6 Cauchy-Matrix 27

2.7 Matrixexponentialfunktion 27

2.8 Padé-Approximation 28

2.9 Gerschgorin-Kreise 28

2.10 Numerischer Rang 28

3 Aufdatierung der LU-Faktorisierung **31**

3.1 Motivation 31

3.2 Problemdefinition 32

3.3 Algorithmus I 33

3.4 Zeilenweise Aufdatierung 34

3.5 Algorithmus II 36

3.6 Aufdatierung mit Spaltenpermutationen 39

3.7 Algorithmus III . . . 41
3.8 Anwendung . . . 44
3.8.1 Aufdatierung der Ableitungsmatrizen . . . 45
3.8.2 Nullraumdarstellung . . . 46
3.8.3 Aufdatierung der transformierten Hessematrix . . . 47
3.9 Numerische Ergebnisse . . . 50
3.9.1 Allgemeine Resultate . . . 51
3.9.2 KKT-Anwendung . . . 54
3.10 Zusammenfassung . . . 55

4 Aufdatierung der Singulärwertzerlegung 57
4.1 Motivation . . . 57
4.2 Symmetrisches Eigenwertproblem . . . 58
4.2.1 Reduktion der Dimension . . . 59
4.2.2 Berechnung der aufdatierten Eigenwerte . . . 61
4.2.3 Berechnung der aufdatierten Eigenvektoren . . . 63
4.3 Multiplikation mit Cauchy-Matrizen . . . 63
4.3.1 Der erweiterte GGS-Algorithmus . . . 63
4.3.2 Transformation von Cauchy-Matrizen . . . 65
4.3.3 Effiziente Multiplikation durch Matrixapproximation . . . 66
4.3.4 Approximation durch Exponentialsummen . . . 73
4.4 Aufdatierung im unsymmetrischen Fall . . . 77
4.5 Aufdatierungsverfahren . . . 78
4.5.1 Übergang zur symmetrischen Darstellung . . . 78
4.5.2 Aufdatierung der Singulärwerte . . . 81
4.5.3 Berechnung der Cauchy-Matrix . . . 82
4.5.4 Aufdatierung der Singulärvektoren . . . 84
4.5.5 Korrekturschritt . . . 85
4.5.6 Zusammenfassung des SVD Aufdatierungsalgorithmus . . . 88
4.6 Numerische Ergebnisse . . . 89
4.7 Zusammenfassung . . . 94

5 Aufdatierung der Matrixexponentialfunktion 97
5.1 Motivation . . . 97
5.2 Problemdefinition . . . 98
5.3 Aufdatierung mittels Cauchy-Integral . . . 99

5.3.1 Numerische Integration 100
5.3.2 Effiziente Berechnung 104
5.3.3 Anwendung auf die Matrixexponentialfunktion 109
5.4 Der Scaling und Squaring Algorithmus 110
5.5 Aufdatierung mittels Scaling und Squaring 115
5.6 Numerische Ergebnisse 120
5.6.1 Verwendung des Cauchy-Integrals 120
5.6.2 Verwendung von Scaling und Squaring 130
5.6.3 Anwendungsbeispiel 133
5.7 Zusammenfassung 138

6 Zusammenfassung **139**

Kapitel 1

Einleitung

1.1 Ziel der Arbeit

Zur Lösung zahlreicher fundamentaler Problemstellungen in der Numerik sind Algorithmen von hoher Komplexität erforderlich. Zwei elementare Problemstellungen sind beispielsweise die Berechnung des Rangs einer Matrix sowie das Lösen linearer Gleichungssysteme. Zur Lösung dieser Probleme können typischerweise Matrixfaktorisierungen verwendet werden. Für die hier genannten Aufgaben wären dies zum einen die Singulärwertzerlegung, welche zur Berechnung des Rangs einer Matrix einen Rechenaufwand von $\frac{4}{3}n^3 + \mathcal{O}(n^2)$ Operationen für eine Matrix der Dimension $(n \times n)$ erfordert, siehe [29]. Zum anderen erfordert auch die direkte Lösung linearer Gleichungssysteme mit Hilfe der LU-Zerlegung einen Rechenaufwand von $\frac{2}{3}n^3$ Operationen. Sind diese Aufgaben in praktischen Anwendungen für dichtbesetzte Matrizen sehr großer Dimension zu lösen oder tritt sogar eine Folge dieser Probleme auf, erreichen auch moderne Computer ihre Leistungsgrenzen.

Ein Beispiel einer solchen Aufgabenstellung ist das Lösen einer Folge von k Gleichungssystemen, deren Systemmatrizen sich jeweils nur durch einen Rang-1 Term unterscheiden. Das bedeutet für eine explizite Neuberechnung der LU-Zerlegungen

$$A_1 = L_1 U_2,\ A_2 = L_2 U_2, \cdots, A_k = L_k U_k \tag{1.1}$$

wäre ein Rechenaufwand von $\mathcal{O}(kn^3)$ Operationen nötig. Hieraus resultiert die Frage, ob es möglich ist, aufgrund der Niedrigrangänderung der einzelnen Matrizen

$$A_2 = A_1 + a_1 b_1^T,\ A_3 = A_2 + a_2 b_2^T, \cdots, A_k = A_{k-1} + a_{k-1} b_{k-1}^T, \tag{1.2}$$

welche durch die Vektoren a_i und b_i erfolgt, die jeweiligen neuen LU-Zerlegungen direkt aus den Vorgängerwerten zu bestimmen. In der Tat existieren Algorithmen, mit welchen ohne die explizite Berechnung der Matrix A_{i+1} die LU-Zerlegung $A_{i+1} = L_{i+1}U_{i+1}$ direkt aus der Faktorisierung $A_i = L_iU_i$ bestimmt werden kann. Hierzu ist im einfachsten Fall ein Rechenaufwand von nur $4n^2$ Operationen nötig, siehe [6]. Ist die initale Faktorisierung $A_1 = L_1U_1$ bekannt, so ist demnach die Folge von LU-Zerlegungen

$$L_2U_2,\ L_3U_3, \cdots, L_kU_k \tag{1.3}$$

mit dieser Vorgehensweise in $\mathcal{O}((k-1)n^2)$ Operationen berechenbar.

Das zentrale Thema dieser Arbeit ist die Entwicklung effizienter Algorithmen zur Durchführung von Rang-1 Aufdatierungen in verschiedenen Problemstellungen. Das primäre Ziel ist dabei, die Größenordung der zur Berechnung der aufdatierten Werte benötigten Operationen signifikant zu reduzieren. Der zweite Schwerpunkt ist die Gewährleistung der numerischen Stabilität der neuen Verfahren. Im Speziellen werden folgende Probleme betrachtet:

- Die Rang-1 Aufdatierung der LU-Faktorisierung mit vollständiger Pivotisierung sowie daraus folgende Niedrigrangkorrekturen in der Anwendung im Optimierungspaket LRAMBO, vgl. [36].
- Die Rang-1 Aufdatierung der Singulärwertzerlegung. Hierbei liegt der Schwerpunkt auf der effizienten Multiplikation von Matrizen unter speziellen strukturellen Voraussetzungen.
- Die Rang-1 Aufdatierung der Matrixexponentialfunktion. Dabei werden Algorithmen zur Aufdatierung mit Hilfe des Cauchy-Integrals sowie per Scaling und Squaring Verfahren vorgestellt.

1.2 Struktur der Arbeit

Zunächst werden in Kapitel 2 die für die nachfolgenden Ausführungen wichtigsten Grundlagen in kurzer Form wiedergegeben. Hierbei sind zu jedem Thema Quellen verzeichnet, in welchen die jeweiligen Probleme ausführlich behandelt werden.

Im Weiteren ist diese Arbeit in drei große Themengebiete unterteilt. In Kapitel 3 wird die Rang-1 Aufdatierung der LU-Faktorisierung behandelt. Nach der einführenden Darstellung einiger bereits bekannter Algorithmen werden die neu entwickelten Verfahren ausführlich beschrieben. Im Folgenden wird deren Anwendung im Optimierungspaket

LRAMBO dargestellt. Abschließend wird anhand von numerischen Ergebnissen die Effizienz als auch die Genauigkeit dieser Verfahren verifiziert.

In Kapitel 4 wird die Rang-1 Aufdatierung der Singulärwertzerlegung betrachtet. Hierbei wird zunächst dargestellt, wie das allgemeine Problem in eine symmetrische Darstellung überführt werden kann. Daraufhin ergibt sich als Schwerpunktthema des Verfahrens die effiziente Matrix-Matrix Multiplikation. Verschiedene Vorgehensweisen, wie diese unter Ausnutzung der Cauchy-Matrix Struktur durchgeführt werden kann, sind in Abschnitt 4.3 gegeben. Der besondere Augenmerk liegt in diesem Zusammenhang auf einer geeigneten Niedrigrangapproximation der Cauchy-Matrix. Nachfolgend wird ein Korrekturschritt beschrieben welcher eine hohe Genauigkeit der aufdatierten Matrizen gewährleistet. Abschließend wird der neu entwickelte Algorithmus auf verschiedene Probleme angewendet. Die dabei erzielten numerischen Ergebnisse werden in Abschnitt 4.6 wiedergegeben.

Thema von Kapitel 5 ist die Rang-1 Aufdatierung der Matrixexponentialfunktion. Hierzu wird zunächst beschrieben, wie die aufdatierte Matrixexponentialfunktion als Cauchy-Integral dargestellt werden kann. Darauf folgend wird erläutert, wie das zur Aufdatierung benötigte Integral numerisch berechnet werden kann. Hierzu wird eine ausführliche Betrachtung des Integrationsfehlers durchgeführt. Weiterhin werden einige Ideen gegeben, wie die numerisch aufwendigen Schritte der Aufdatierung effizient berechnet werden können. Im Weiteren wird der Scaling und Squaring Algorithmus zur Berechnung der Matrixexponentialfunktion beschrieben. Wie dieser Algorithmus zur Lösung des Aufdatierungsproblems genutzt werden kann, stellt einen weiteren Schwerpunkt dieses Kapitels dar. Abschließend werden die zuvor beschriebenen Aufdatierungsverfahren an zahlreichen numerischen Beispielen getestet.

Die erzielten Ergebnisse aller Themengebiete werden abschließend in Kapitel 6 zusammengefasst.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für die Arbeit wichtigsten Grundlagen wiederholt. Dies sind zunächst die LU-Faktorisierung, die Singulärwertzerlegung sowie die Matrixexponentialfunktion, deren Rang-1 Aufdatierungen das zentrale Thema dieser Arbeit sind. Die dazu am häufigsten verwendeten Techniken, wie beispielsweise verschiedene Matrixtransformationen, werden im Folgenden ebenfalls dargestellt. Weitere, im späteren Verlauf wichtige Themen sind unter anderem die Cauchy-Matrix, das Cauchy-Integral sowie die Padé-Approximation. Einige Grundlagen hierzu werden nachfolgend in kurzer Form wiedergegeben. Weiterführende Darstellungen sind in den jeweils genannten Quellen gegeben.

2.1 Orthogonale Transformationen

2.1.1 Givens-Rotationen

Mit Givens-Drehungen wird eine bestimmte Klasse elementarer, orthogonaler Transformationsmatrizen bezeichnet. Diese sind definiert durch:

$$G_{i,j} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \ldots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \ldots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{2.1}$$

mit

$$c^2 + s^2 = 1. \tag{2.2}$$

Matrizen der Art (2.1) unterscheiden sich von der Einheitsmatrix nur in jeweils zwei Elementen der i-ten und j-ten Zeile. Die Orthogonalität von G_{ij} folgt sofort aus Bedingung (2.2).

Bei der Anwendung einer Givens-Drehung $G_{i,j} \in \mathbb{R}^{n\times n}$ auf einen Vektor $v \in \mathbb{R}^n$ ändern sich in diesem genau die beiden Elemente v_i und v_j. Es ist in diesem Fall

$$G_{i,j}v = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \dots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \dots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \vdots \\ v_i \\ \vdots \\ v_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ cv_i + sv_j \\ \vdots \\ cv_j - sv_i \\ \vdots \end{bmatrix}.$$

Geometrisch betrachtet handelt es sich hierbei um eine Drehung des Vektors v in der von den Einheitsvektoren e_i und e_j aufgespannten Ebene um den Winkel ϕ. Dabei ist ϕ durch $c = \cos(\phi)$ bzw. $s = \sin(\phi)$ definiert.

Givens-Rotationen werden typischerweise zur gezielten Elimination einzelner Elemente in Vektoren bzw. Matrizen verwendet um somit gewünschte Strukturen herzustellen. Eine stabile Möglichkeit zur Bestimmung von c und s für diesen Fall ist unter anderem in [29] gegeben.

2.1.2 Householder-Spiegelungen

Householder-Spiegelungen sind eine Klasse elementarer, orthogonaler Transformationen. Diese sind wie folgt definiert:

Definition 2.1.1. *Sei der Vektor $v \in \mathbb{R}^n \neq 0$. Dann ist durch*

$$H = I - \frac{2}{v^T v} v v^T \tag{2.3}$$

eine Householder-Spiegelung gegeben. Diese ist eine symmetrische, orthogonale Rang-1 Modifikation der Einheitsmatrix.

Die Multiplikation einer durch $v \in \mathbb{R}^n$ definierten Householder-Matrix $H \in \mathbb{R}^{n \times n}$ mit einem Vektor $y \in \mathbb{R}^n$ führt zu

$$Hy = \left(I - \frac{2vv^T}{v^T v} \right) y = y - \frac{2v^T y}{v^T v} v. \tag{2.4}$$

Dies entspricht geometrisch betrachtet einer Spiegelung des Vektors y an der zu v orthogonalen Hyperebene.

Householder-Spiegelungen der Art (2.4) werden typischerweise verwendet, um Vektoren y auf ein Vielfaches des ersten Einheitsvektors zu transformieren. Dazu muss

$$v = y \pm \|y\|_2 e_1$$

gewählt werden. Weitere Details zur stabilen Berechnung von Householder-Matrizen sind unter anderem in [72] gegeben.

2.1.3 Jacobi-Methode

Ziel der Jacobi-Methode ist es, die Außerdiagonalelemente einer symmetrischen Matrix vom Betrag zu verkleinern und diese somit näher an Diagonalform zu bringen. Dazu werden Jacobi-Rotationen der Form

$$J_{i,j} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \ldots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \ldots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{2.5}$$

verwendet. $J_{i,j}$ unterscheidet sich nur in den Elementen (i,i), (i,j), (j,i) und (j,j) von der Einheitsmatrix.

Bemerkung 2.1.1. *Jacobi-Rotationen unterscheiden sich nicht von den in Kapitel 2.1.1 vorgestellten Givens-Drehungen. Sie wurden allerdings im Zusammenhang mit der hier vorgestellten Methode bereits 1864 von Jacobi verwendet [49].*

Sei $A \in \mathbb{R}^{n \times n}$ eine symmetrische Matrix, welche diagonalisiert werden soll. Ein Jacobi Schritt besteht daraus, zunächst ein Indexpaar (i, j) mit $1 \leq i, j \leq n$ auszuwählen und dazu c und s so zu berechnen, dass

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \tilde{a}_{ii} & \tilde{a}_{ij} \\ \tilde{a}_{ji} & \tilde{a}_{jj} \end{bmatrix} \tag{2.6}$$

und $\tilde{a}_{ij} = \tilde{a}_{ji} = 0$ ist. Die neu erhaltene Matrix $\tilde{A}$ ist in dem Sinne näher an Diagonalform als A, dass

$$\sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \tilde{a}_{ij}^2} \leq \sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} a_{ij}^2}$$

ist.

Beweis. Siehe [29]. □

Die Durchführung von n Schritten der Art (2.6) wird typischerweise als eine Sequenz bezeichnet. Es gibt verschiedene Strategien, wie die Indexpaare (i, j) innerhalb einer Sequenz gewählt werden.

Im klassischen Algorithmus wird jeweils das Maximum der Außerdiagonalelemente von A gesucht und (i, j) so gewählt, dass dieses eliminiert wird. Der Nachteil dieses Vorgehens ist, dass zur Suche des Maximums jeweils alle Matrixelemente betrachtet werden müssen.

Eine Alternative dazu ist der zyklische Algorithmus, in welchem die Indexpaare zum Beispiel folgende Ordnung durchlaufen:

$$(i, j) = (1, 2), (1, 3), \ldots, (1, n), (2, 3), (2, 4), \ldots, (2, n), (3, 4), \ldots \quad .$$

Beide Algorithmen liefern eine asymptotisch, quadratische Konvergenz zur Reduktion der Diagonalelemente, siehe [72, 73]. Des Weiteren kann heuristisch argumentiert werden, dass die Anzahl nötiger Sequenzen, um A auf Diagonalgestalt zu reduzieren, proportional zu $\log(n)$ ist, siehe [12].

Die Jacobi-Methode ist besonders effizient, wenn nur wenige Elemente der Matrix A zu eliminieren sind, siehe [75].

2.2 LU-Faktorisierung

Eine Matrix $A \in \mathbb{R}^{n \times n}$ besitzt genau dann eine LU-Zerlegung

$$PA = LU, \tag{2.7}$$

wenn $\det(A) \neq 0$ ist. Hierbei bezeichnet $U \in \mathbb{R}^{n\times n}$ eine reguläre, obere Dreiecksmatrix, $L \in \mathbb{R}^{n\times n}$ eine untere Dreiecksmatrix, deren Diagonalelemente gleich eins sind und $P \in \mathbb{R}^{n\times n}$ eine Permutationsmatrix. (2.7) wird als LU-Faktorisierung mit partieller Pivotisierung bezeichnet, siehe [8]. Die Berechnung dieser Zerlegung erfolgt durch den Gauß-Algorithmus [29] und erfordert $\frac{2}{3}n^3$ Operationen. Die LU-Faktorisierung wird typischerweise zur Lösung linearer Gleichungssysteme $Ax = b$ verwendet.

Die LU-Zerlegung mit partieller Pivotisierung einer rechteckigen Matrix $A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \in \mathbb{R}^{n\times m}$, mit $m > n$ und $A_1 \in \mathbb{R}^{n\times n}$ sowie $A_2 \in \mathbb{R}^{n\times m-n}$ existiert genau dann, wenn $\det(A_1) \neq 0$ ist. Mit der zusätzlichen Verwendung von Spaltenpermutationen existiert zu jeder Matrix $A \in \mathbb{R}^{n\times m}$, welche vollen Rang hat, die Faktorisierung der Form

$$PAQ = LU. \tag{2.8}$$

Hierbei ist $Q \in \mathbb{R}^{m\times m}$ eine Spaltenpermutationsmatrix und P, L sowie U wie bereits zuvor definiert. (2.8) wird als LU-Zerlegung mit vollständiger Pivotisierung bezeichnet.

Für weitere Details siehe [29].

2.3 Singulärwertzerlegung

Ein zentrales Thema dieser Arbeit ist die in Kapitel 4 beschriebene Rang-1 Aufdatierung der Singulärwertzerlegung. Aus diesem Grund soll an dieser Stelle zunächst der Algorithmus zur Berechnung dieser Faktorisierung im Detail beschrieben werden. Der dazu benötige, hohe Berechnungsaufwand verdeutlicht die Notwendigkeit effizienter Aufdatierungsalgorithmen.

Gegeben sei die Matrix $A \in \mathbb{R}^{m\times n}$, mit $m \leq n$. Dann existieren orthogonale Matrizen $U \in \mathbb{R}^{m\times m}$ und $V \in \mathbb{R}^{n\times n}$, so dass

$$U^T AV = \begin{bmatrix} \Sigma & 0 \end{bmatrix}, \tag{2.9}$$

wobei $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$ mit $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$. Die Faktorisierung

$$A = U \begin{bmatrix} \Sigma & 0 \end{bmatrix} V^T \tag{2.10}$$

wird als Singulärwertzerlegung (SVD) von A bezeichnet. Die Elemente σ_i werden Singulärwerte von A genannt. Gilt für eine Anzahl von r Singulärwerten, dass $\sigma_i = \sigma_{i+1} = \cdots = \sigma_{i+r-1}$, so spricht man von einer Vielfachheit r des Singulärwertes σ_i.

Die Spalten von $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}$ und $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$ werden als linke- bzw. rechte Singulärvektoren von A bezeichnet. Diese erfüllen folgende Bedingungen:

$$\begin{aligned} Av_i &= \sigma_i u_i, \quad i = 1, \ldots, m, \\ A^T u_i &= \sigma_i v_i, \quad i = 1, \ldots, m, \\ Av_i &= 0, \quad i = m+1, \ldots, n. \end{aligned}$$

Die Singulärwerte σ_i der Zerlegung (2.9) sind eindeutig. Die rechten Singulärvektoren, welche zu einem einfachen Singulärwert gehören, sind bis auf das Vorzeichen ebenfalls eindeutig. Die zu einem Singulärwert mit Vielfachheit größer eins gehörigen Singulärvektoren spannen einen eindeutig definierten Raum auf. Die Singulärvektoren selbst sind in diesem Fall nicht eindeutig. Die linken Singulärvektoren sind mittels $Av_i = \sigma_i u_i$ durch die zugehörigen rechten Singulärvektoren und Singulärwerte eindeutig bestimmt, sofern $\sigma_i \neq 0$ gilt.

Die Berechnung der Singulärwertzerlegung erfolgt in mehreren Schritten. Zunächst wird die ursprüngliche Matrix A mittels orthogonaler Transformationen auf die Bidiagonalform $B \in \mathbb{R}^{m \times n}$ reduziert. Im weiteren werden dann die Superdiagonalelemente von B eliminiert. Im Folgenden wird dieser Algorithmus zusammengefasst dargestellt. Weitere Details sind unter anderem [28, 29, 68] zu entnehmen.

Reduktion auf Bidiagonalform

Die Reduktion auf Bidiagonalform soll an Hand von

$$A = \begin{bmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \end{bmatrix} \in \mathbb{R}^{4 \times 6}$$

schematisch dargestellt werden. Durch die Multiplikation von A mit orthogonalen Drehungs- bzw. Spiegelungsmatrizen können an den gewünschten Stellen in A Nullen erzeugt werden. Im folgenden Schema werden zunächst Householderspiegelungen, vgl. Kapitel 2.1, $U_i \in \mathbb{R}^{4 \times 4}$ und $V_i \in \mathbb{R}^{6 \times 6}$ mit $i = 1, \ldots 3$ verwendet, um durch deren abwechselnde

Multiplikation von links bzw. rechts A auf Bidiagonalform zu transformieren.

$$A \xrightarrow{U_1\cdot} \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \end{bmatrix} \xrightarrow{\cdot V_1} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \end{bmatrix}$$

$$\xrightarrow{U_2\cdot} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & X & X & X & X \end{bmatrix} \xrightarrow{\cdot V_2} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & X & X \\ 0 & 0 & X & X & X & X \end{bmatrix}$$

$$\xrightarrow{U_3\cdot} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \end{bmatrix} \xrightarrow{\cdot V_3} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & X & X & X \end{bmatrix}$$

Da die Bidiagonalform B nur im vorderen (4×4)-Block Nichtnullelemente haben soll, müssen nachfolgend noch die Elemente $(4,5)$ sowie $(4,6)$ eliminiert werden. Dies geschieht durch eine Folge von Givensrotationen, vgl. Kapitel 2.1, von rechts $V_j \in \mathbb{R}^{6\times 6}$, $j = 4, \ldots, 8$:

$$\xrightarrow{\cdot V_4} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & X & X & 0 \end{bmatrix} \xrightarrow{\cdot V_5} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & X & 0 \\ 0 & 0 & 0 & X & 0 & 0 \end{bmatrix}$$

$$\xrightarrow{\cdot V_6} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & X & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & X & 0 & 0 \end{bmatrix} \xrightarrow{\cdot V_7} \begin{bmatrix} X & X & 0 & 0 & X & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & X & 0 & 0 \end{bmatrix}$$

$$\xrightarrow{\cdot V_8} \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & X & 0 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{B} & 0 \end{bmatrix}}_{B}. \tag{2.11}$$

Zusammenfassend ergibt sich in diesem Beispiel

$$B = \underbrace{U_3 \cdots U_1}_{\tilde{U}^T} A \underbrace{V_1 \cdots V_8}_{\tilde{V}}. \tag{2.12}$$

Aufgrund der Orthogonalität der Matrizen U_i sowie V_j sind $\tilde{U}$ und $\tilde{V}$ ebenfalls orthogonal.

Elimination der Superdiagonalelemente

Das verbleibende Problem besteht in der Berechnung der SVD der bidiagonalen Matrix

$$\tilde{B} = \begin{bmatrix} d_1 & u_1 & 0 & \cdots & & 0 \\ 0 & d_2 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & u_{m-1} \\ 0 & \cdots & \cdots & \cdots & 0 & d_m \end{bmatrix}. \tag{2.13}$$

Dies geschieht durch die implizite Anwendung des symmetrischen QR-Algorithmus, vgl. [29], auf $T = \tilde{B}^T\tilde{B}$. Da eine explizite Berechnung des Matrixprodukts $\tilde{B}^T\tilde{B}$ numerische Probleme hervorrufen würde, werden die nötigen orthogonalen Transformationen direkt auf $\tilde{B}$ angewendet. Hierzu wird der stabile, implizite QR-Schritt verwendet. Dabei wird implizit $T - \lambda I = QR$ zerlegt und nachfolgend T durch Q^TTQ ersetzt. Aus dem impliziten Q-Theorem, siehe [29], folgt, dass die folgenden drei Schritte genau dies liefern, sofern $t_{i,i+1} \neq 0$ für $i = 1, \ldots, n-1$ gilt.

(1) Berechnung des Eigenwertes λ, von

$$\tilde{T} = \begin{bmatrix} d_{m-1}^2 + u_{m-2}^2 & d_{m-1}u_{m-1} \\ d_m u_{m-1} & d_m^2 + u_{m-1}^2 \end{bmatrix}, \tag{2.14}$$

welcher näher zu $d_m^2 + u_{m-1}^2$ ist.

(2) Berechnung der Givensrotation G_1, so dass

$$G_1^T \begin{bmatrix} d_1^2 - \lambda \\ d_1 u_1 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}, \quad \alpha \in \mathbb{R}. \tag{2.15}$$

(3) Berechnung der Givens-Rotationen G_2 bis G_{n-1}, so dass

$$\underbrace{G_{m-1}^T \cdots G_1^T}_{Q^T} T \underbrace{G_1 \cdots G_{m-1}}_{Q} \tag{2.16}$$

tridiagonal ist und $Qe_1 = G_1 e_1$.

Um die explizite Formulierung von T zu vermeiden, werden die Givens-Rotationen G_i direkt auf $\tilde{B}$ angewendet. Das zuvor betrachtete Beispiel lässt sich somit wie folgt fortsetzen:

$$\tilde{B} \leftarrow \tilde{B}G_1 = \begin{bmatrix} X & X & 0 & 0 \\ X & X & X & 0 \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{bmatrix}. \tag{2.17}$$

Vor der weiteren Multiplikation mit den Givens-Rotationen $G_2, \ldots, G_{m-1}$ muss zunächst das neu entstandene Nicht-Null-Element $\tilde{B}_{2,1}$ eliminiert werden. Auch dies geschieht durch die Anwendung orthogonaler Transformationen $\tilde{U}_i$, und $\tilde{V}_i$ von links bzw. rechts:

$$\begin{aligned}
&\overset{\tilde{U}_1^T\cdot}{\rightarrow} \begin{bmatrix} X & X & X & 0 \\ 0 & X & X & 0 \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{bmatrix} \quad \overset{\cdot\tilde{V}_1}{\rightarrow} \begin{bmatrix} X & X & 0 & 0 \\ 0 & X & X & 0 \\ 0 & X & X & X \\ 0 & 0 & 0 & X \end{bmatrix} \\
&\overset{\tilde{U}_2^T\cdot}{\rightarrow} \begin{bmatrix} X & X & 0 & 0 \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{bmatrix} \quad \overset{\cdot\tilde{V}_2}{\rightarrow} \begin{bmatrix} X & X & 0 & 0 \\ 0 & X & X & 0 \\ 0 & 0 & X & X \\ 0 & 0 & X & X \end{bmatrix} \\
&\overset{\tilde{U}_3^T\cdot}{\rightarrow} \begin{bmatrix} X & X & 0 & 0 \\ 0 & X & X & 0 \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{bmatrix} = \bar{B}.
\end{aligned} \tag{2.18}$$

Dieser Vorgang erzeugt die neue Bidiagonalmatrix

$$\bar{B} = (\underbrace{U_{m-1}^T \cdots U_1^T}_{\bar{U}^T}) B (\underbrace{G_1, V_1 \cdots V_{m-2}}_{\bar{V}}). \tag{2.19}$$

Numerische Beobachtungen, siehe [28], haben gezeigt, dass nur wenige Iterationen der Art (2.18) benötigt werden, um numerisch vernachlässigbar kleine Außerdiagonaleinträge u_i zu erhalten. Ist $\bar{B}$ in Diagonalform, so sind seine Diagonalelemente gleich den Singulärwerten σ_i von A. Demzufolge ist dann $\Sigma = B$. Die orthogonalen Matrizen U und V aus (2.10) ergeben sich durch die Aufmultiplikation der jeweils rechtsseitigen- bzw. linksseitigen Transformationen in (2.12) und (2.19).

Der Gesamtaufwand zur Berechnung der Singulärwertzerlegung beträgt $4mn^2 - \frac{4}{3}n^3$ Operationen für den Fall, dass nur die Matrix Σ benötigt wird und $4m^2n + 8mn^2 + 9n^3$ Operationen, wenn zusätzlich auch U und V explizit berechnet werden sollen.

Bemerkung 2.3.1. *Eine Iteration der Form (2.18) ist nur möglich, sofern weder d_i noch u_i in (2.13) gleich Null sind. Tritt ein solcher Fall ein, wird die Matrix $\tilde{B}$ durch weitere orthogonale Transformationen in*

$$\tilde{B} = \begin{bmatrix} \tilde{B}_1 & 0 \\ 0 & \tilde{B}_2 \end{bmatrix} \tag{2.20}$$

entkoppelt. Sind u_i bzw. d_i vernachlässigbar klein, so werden diese gleich Null gesetzt, was ebenfalls zu einer Entkopplung (2.20) des Problems führt. Daraufhin kann die Berechnung mit den einzelnen Submatrizen aus (2.20) separat fortgestzt werden.

2.4 Sherman-Morrison-Woodbury Formel

Seien $A \in \mathbb{R}^{n\times n}$ und $U, V \in \mathbb{R}^{n\times k}$, dann ist die Inverse von $A + UV^T$ durch

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1} \tag{2.21}$$

gegeben. Hierfür sei angenommen, dass sowohl A als auch $A + UV^T$ nicht-singulär sind. Aus

$$\det(A + UV^T) = \det(A) \cdot \det(I + V^TA^{-1}U)$$

folgt in diesem Fall die Nicht-Singularität von $I + V^TA^{-1}U$. Gleichung (2.21) wird als Sherman-Morrison-Woodbury Formel bezeichnet, [29].

2.5 Cauchy-Integral

Sei $A \in \mathbb{C}^{n\times n}$, so gilt

$$f(A) = \frac{1}{2\pi i}\int_\Gamma f(z)(zI - A)^{-1}dz,$$

für Funktionen f, welche innerhalb sowie auf der Kurve Γ analytisch sind. Γ ist dabei eine einfach geschlossene Kurve, die das Spektrum $\sigma(A)$ von A umschließt.

2.6 Cauchy-Matrix

Definition 2.6.1. *Es seien $s \in \mathbb{R}^n$, $t \in \mathbb{R}^n$ zwei Vektoren mit disjunkten Elementen, d.h. $t_i \neq s_j$ für alle $i, j = 1, \ldots n$, sowie $p \ll n$. Dann wird mit*

$$(B_p)_{i,j} = \frac{1}{(t_i - s_j)^p}, \quad i, j = 1, \ldots, n \tag{2.22}$$

die verallgemeinerte Hilbertmatrix $B_p \in \mathbb{R}^{n \times n}$ bezeichnet. Die im Spezialfall $p = 1$ entstehende Matrix

$$(C)_{i,j} = \frac{1}{t_i - s_j}, \quad i, j = 1, \ldots, n \tag{2.23}$$

wird als Cauchy-Matrix $C_{(t,s)} \in \mathbb{R}^{n \times n}$ bezeichnet.

2.7 Matrixexponentialfunktion

Definition 2.7.1. *Sei $A \in \mathbb{C}^{n \times n}$, dann ist durch*

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \tag{2.24}$$

die Matrixexponentialfunktion $e^A \in \mathbb{C}^{n \times n}$ definiert.

Zahlreiche weitere Definitionen der Exponentialfunktion für matrixwertige Argumente sind in [47, 58, 59] zu finden. Unter anderem kann die Matrixexponentialfunktion mit Hilfe des Cauchy-Integrals

$$e^A = \frac{1}{2\pi i} \int_\Gamma e^z (zI - A)^{-1} dz,$$

als der Grenzwert

$$e^A = \lim_{s \to \infty} \left(I + \frac{A}{s} \right)^s,$$

als Lösung der Differentialgleichung

$$Y'(t) = AY(t), \quad Y(0) = I,$$

oder als Padé-Approximierte

$$e^A \approx p_{km}(A) \cdot q_{km}(A)^{-1} \tag{2.25}$$

dargestellt werden.

2.8 Padé-Approximation

Definition 2.8.1. *Mit $\mathcal{R}_n$ sei der Raum der Polynome n-ten Grades bezeichnet. Die rationale Funktion*

$$r_{km} = \frac{p_{km}}{q_{km}}$$

ist die $[k/m]$ Padé-Approximation einer Funktion $f(x)$, wenn $p_{km} \in \mathcal{R}_k$, $q_{km} \in \mathcal{R}_m$, $q_{km}(0) = 1$ sowie

$$f(x) - r_{km}(x) = \mathcal{O}(x^{k+m+1})$$

gilt. Typischerweise wird verlangt, dass p_{km} und q_{km} keine gemeinsamen Nullstellen haben.

Existiert eine $[k/m]$ Padé-Approximation von f, so ist diese eindeutig, siehe [5].

2.9 Gerschgorin-Kreise

Mit Hilfe von Gerschgorin-Kreisen [25] können einfache Gebiete angegeben werden, welche die Eigenwerte einer Matrix enthalten.

Definition 2.9.1. *Sei $A = (a_{ij}) \in \mathbb{C}^{n\times n}$. Dann ist der i-te Gerschgorin-Kreis durch*

$$K_i = K\left(a_{ii}, \sum_{j=1, j\neq i}^{n} |a_{ij}|\right)$$

definiert. Hierbei ist $K(m, r)$ ein abgeschlossener Kreis mit Mittelpunkt m und Radius r.

Das Spektrum der Matrix A ist eine Teilmenge der Vereinigung $G_1 = \bigcup_{i=1}^{n} K_i$. Da $\sigma(A) = \sigma(A^T)$ gilt, lässt sich durch die Vereinigung der zu A^T zugehörigen Gerschgorin-Kreise ein zweites Gebiet G_2 angeben, welches das Spektrum von A enthält. Die Eigenwerte von A befinden sich im Schnitt beider Gebiete $G_1 \cap G_2$.

2.10 Numerischer Rang

Der Rang einer Matrix ist gleich der Anzahl ihrer Singulärwerte, welche von Null verschieden sind. Der numerische Rang einer Matrix $A \in \mathbb{R}^{n\times n}$ wird dagegen mit Hilfe einer Schranke $\delta > 0$ wie folgt definiert, vgl. [3]:

$$\text{rang}_\delta(A) = k, \quad \text{so dass } \sigma_k(A) > \delta \text{ und } \sigma_{k+1}(A) \leq \delta.$$

Dabei bezeichnen $\sigma_i(A)$, $i = 1, \ldots, n$ die der Größe nach absteigend sortierten Singulärwerte von A. Eine typische Wahl der Toleranz ist

$$\delta = \epsilon \|A\|_2 = \epsilon \sigma_1(A),$$

wobei durch ϵ die Maschinengenauigkeit bezeichnet ist.

Kapitel 3

Aufdatierung der LU-Faktorisierung

3.1 Motivation

Eines der am meisten verwendeten direkten Lösungsverfahren für lineare Gleichungssysteme ist der Gauß-Algorithmus. Die damit verbundene Berechnung der LU-Faktorisierung einer $(n \times n)$-Matrix erfordert einen hohen numerischen Rechenaufwand von $\mathcal{O}(n^3)$ Operationen.

Die Lösung nichtlinearer Optimierungsprobleme durch das Paket LRAMBO [34, 36] erfordert die Lösung einer Folge großer, dichtbesetzter Gleichungssysteme. Die Systemmatrix jedes dieser k Gleichungssysteme unterscheidet sich vom vorherigen System allerdings nur durch eine Rang-1 Korrektur. Die explizite Berechnung und Neuzerlegung jedes dieser Systeme würde demnach zu einem Rechenaufwand von jeweils $\mathcal{O}(n^3)$ Operationen führen und wäre äußerst ineffizient.

Im Folgenden werden zunächst einige Verfahren vorgestellt, welche es ermöglichen, die bereits bekannten Matrizen aus der vorherigen Zerlegung bezüglich der Rang-1 Korrektur direkt aufzudatieren. Ein solches Vorgehen ist in einer Komplexität von $\mathcal{O}(n^2)$ möglich. Im Weiteren liegt der Schwerpunkt darauf, diese Aufdatierungsverfahren an die spezielle Anwendung im bereits genannten Optimierungspaket anzupassen und für spezielle Anforderungen zu erweitern.

Von besonderem Interesse ist dabei ein Algorithmus, welcher die Rang-1 Aufdatierung der LU-Faktorisierung mit vollständiger Permutation ermöglicht. Diese ist für die stabile Zerlegung rechteckiger Matrizen unabdingbar. Weiterhin ist aufgrund der sich

ändernden Struktur der Rang-1 Terme während des Lösungsprozesses ein Algorithmus gefragt, welcher sowohl eine möglichst effiziente als auch eine numerisch stabile Aufdatierung ermöglicht. Hierzu werden verschiedene Methoden der Aufdatierung zu einem hybriden Algorithmus zusammengeführt.

Zusätzlich zur Aufdatierung der LU-Zerlegung ist eine damit einhergehende effiziente Modifikation der Bild- sowie Nullraumbasis der Matrix nötig. Daraus resultieren weitere Niedrigrangaufdatierungen verschiedener in LRAMBO auftretender Transformationen. Wie diese speziell im Hinblick auf Spaltenpermutationen effizient zu berechnen sind, wird ebenfalls dargestellt.

Die hier beschriebenen Ergebnisse stellen eine Erweiterung zu [64] dar und wurden in [67] veröffentlicht.

3.2 Problemdefinition

Gegeben sei die LU-Zerlegung der Matrix $A \in \mathbb{R}^{m \times n}, m \leq n$, mit vollständiger Permutation

$$PAQ = LU \quad . \tag{3.1}$$

Dabei ist $P \in \mathbb{R}^{m \times m}$ eine Zeilen- sowie $Q \in \mathbb{R}^{n \times n}$ eine Spaltenpermutationsmatrix. $L \in \mathbb{R}^{m \times m}$ ist eine untere Dreiecksmatrix deren Hauptdiagonalelemente gleich eins sind. Die rechteckige Matrix

$$U = \left[\underbrace{U_1}_{\in \mathbb{R}^{m \times m}} \quad \underbrace{U_2}_{\in \mathbb{R}^{m \times d}} \right] \in \mathbb{R}^{m \times n} \tag{3.2}$$

sei unterteilt in eine obere Dreiecksmatrix U_1 sowie einen vollbesetzten hinteren Teil U_2, $d = n - m$.

Des Weiteren sei die Matrix $A_+ \in \mathbb{R}^{m \times n}$ definiert durch

$$A_+ = A + uv^T, \tag{3.3}$$

wobei die Vektoren $u \in \mathbb{R}^m$ sowie $v \in \mathbb{R}^n$ eine Rang-1 Modifikation zu A definieren. Ziel ist die effiziente Berechnung einer LU-Zerlegung der aufdatierten Matrix

$$P_+ A_+ Q_+ = L_+ U_+, \tag{3.4}$$

wobei die Matrizen L_+, U_+, P_+ sowie Q_+ analog zu L, U, P bzw. Q definiert sind. Auf die explizite Berechnung der Summe $A + uv^T$ und eine folgende Neufaktorisierung soll dabei verzichtet werden, da dies einen Rechenaufwand von $\mathcal{O}(nm^2)$ mit sich führen

würde, vgl. [17]. Stattdessen sollen zur Berechnung von (3.4) die bereits vorhandenen Faktoren aus (3.1) direkt modifiziert werden. Wie im Weiteren gezeigt wird, ist dies mit einem numerischen Aufwand von $\mathcal{O}(mn)$ möglich.

Im Folgenden werden zunächst zwei bereits bekannte Verfahren, welche die effiziente Aufdatierung der LU-Zerlegung ohne bzw. nur mit Zeilenpermutation ermöglichen, vorgestellt. Im Weiteren wird ein neu entwickelter Algorithmus dargestellt, der die Vorteile beider Verfahren kombiniert sowie um die Möglichkeit der Spaltenpermutation erweitert. Dieses neue, hybride Verfahren stellt eine Erweiterung der Arbeit [64] dar. Abschließend werden numerische Ergebnisse der verschiedenen Methoden sowohl allgemein als auch bei deren Anwendung im Optimierungspaket LRAMBO vorgestellt.

3.3 Algorithmus I

Der Algorithmus von Bennett [6] ermöglicht die effiziente Berechnung der LU-Faktorisierung einer bereits zerlegten Matrix A nach Rang-1 Aufdatierung ohne Permutation, das heißt

$$\underbrace{L_+U_+}_{A_+} = \underbrace{LU}_{A} + uv^T. \tag{3.5}$$

Zur Veranschaulichung der Funktionsweise dieses Verfahrens eignet sich folgende Blockdarstellung der Zerlegung von A,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ l_{21} & I \end{bmatrix}}_{L_1} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A} \end{bmatrix} \underbrace{\begin{bmatrix} u_{11} & u_{12} \\ 0 & I \end{bmatrix}}_{U_1}, \tag{3.6}$$

wobei $\tilde{A} = A_{22} - l_{21}u_{12}$ ist. Des Weiteren ist L_1 eine Einheitsmatrix, welche in der ersten Spalte um l_{21} erweitert wurde. Analog sei auch U_1, auf die erste Zeile von U bezogen, definiert. Mit dieser Notation kann der Aufdatierungsprozess wie folgt dargestellt werden. Es sei zunächst

$$A_+ = A + \gamma uv^T = L_1 \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A} \end{bmatrix} U_1 + \gamma \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}^T \quad . \tag{3.7}$$

Mit Hilfe der Zerlegung (3.6) in den entsprechenden Dimensionen $l_{21}^+ \in \mathbb{R}^{m-1}$, $u_{11}^+ \in \mathbb{R}$, $u_{12}^+ \in \mathbb{R}^{n-1}$, lässt sich A_+ als

$$A_+ = \underbrace{\begin{bmatrix} 1 & 0 \\ l_{21}^+ & I \end{bmatrix}}_{L_1^+} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}^+ \end{bmatrix} \underbrace{\begin{bmatrix} u_{11}^+ & u_{12}^+ \\ 0 & I \end{bmatrix}}_{U_1^+} \tag{3.8}$$

schreiben. Hierbei sind unter der Voraussetzung $u_{11}^+ \neq 0$:

$$u_{11}^+ = u_{11} + \gamma u_1 v_1 \quad , l_{21}^+ = \frac{l_{21} u_{11} + \gamma u_2 v_1}{u_{11}^+} \quad , u_{12}^+ = u_{12} + \gamma u_1 v_2^T$$

mit $l_{21} \in \mathbb{R}^{m-1}$, $u_{11} \in \mathbb{R}$, $u_{12}^T \in \mathbb{R}^{n-1}$ durch (3.7) eindeutig definiert.

Die in (3.8) noch weiter zu modifizierende Matrix $\tilde{A}_+ \in \mathbb{R}^{m-1 \times n-1}$ lässt sich durch das in der Dimension reduzierte Rang-1 Aufdatierungsproblem

$$\tilde{A}_+ = \tilde{A} + \tilde{\gamma} \tilde{u} \tilde{v}^T \tag{3.9}$$

darstellen. Dabei sind

$$\begin{aligned} \tilde{\gamma} &= \frac{\gamma}{u_{11}^+} \in \mathbb{R} \\ \tilde{u} &= u_2 - l_{21} u_1 \in \mathbb{R}^{m-1} \\ \tilde{v}^T &= u_{11} v_2^T - v_1 u_{12} \in \mathbb{R}^{n-1}. \end{aligned} \tag{3.10}$$

Die aufdatierten Faktoren L_+ und U_+ von (3.5) lassen sich demzufolge durch die $(m-1)$–fache Wiederholung von (3.7)-(3.10) auf die jeweils reduzierten Probleme berechnen. Der gesamte Rechenaufwand für diesen Algorithmus beträgt $4nm+\mathcal{O}(n)+\mathcal{O}(m)$ Operationen.

3.4 Zeilenweise Aufdatierung

Bei der Durchführung des im vorangehenden Kapitel vorgestellten Algorithmus wird die Matrix U zeilenweise, die Matrix L allerdings spaltenweise modifiziert. Auf modernen Rechnerarchitekturen sind Speicherzugriffe bedeutend zeitintensiver als elementare Operationen. Da beide Matrizen L und U zeilenweise gespeichert sind, ist beim spaltenweisen Durchlaufen der Matrix L kein sequenzieller Zugriff auf die jeweils nacheinander folgenden Elemente möglich. Stattdessen muss um jeweils m Elemente im Speicher hin- und her gesprungen werden. Dies verlangsamt den Algorithmus beträchtlich. Es ist allerdings möglich den ursprünglichen LU-Aufdatierungsalgorithmus von Bennett so umzuformulieren, dass eine zeilenweise Aufdatierung sowohl von U als auch von L möglich ist.

Standard LU-Aufdatierung

```
1:  for i = 1 to m do
2:     //diagonal update
3:     U_ii = U_ii + u_i * v_i
4:     v_i = v_i / U_ii
5:     for j = i + 1 to m do
6:        //L update
7:        u_j = u_j − u_i * L_ji
8:        L_ji = L_ji + v_i * u_j
9:     end for
10:    for j = i + 1 to n do
11:       //U update
12:       U_ij = U_ij + u_i * v_j
13:       v_j = v_j − v_i * U_ij
14:    end for
15: end for
```

Zeilenweise LU-Aufdatierung

```
1:  for i = 1 to m do
2:     for j = 1 to i − 1 do
3:        //delayed L update
4:        u_i = u_i − u_j * L_ij
5:        L_ij = L_ij + v_j * u_i
6:     end for
7:     //diagonal update
8:     U_ii = U_ii + u_i * v_i
9:     v_i = v_i / U_ii
10:    for j = i + 1 to n do
11:       //U update
12:       U_ij = U_ij + u_i * v_j
13:       v_j = v_j − v_i * U_ij
14:    end for
15: end for
```

Abbildung 3.1: Ursprünglicher- sowie zeilenweiser Algorithmus zur Rang-1 Aufdatierung der LU-Faktorisierung nach Bennett

Durch diese, in Abbildung 3.1 dargestellte Modifikation, reduziert sich die Laufzeit des Algorithmus um ein Vielfaches.

In den Abbildungen 3.2 und 3.3 ist das unterschiedliche Fortschreiten des Aufdatierungsprozesses in Bezug auf L und U dargestellt. Der schraffierte Bereich in den Abbildungen markiert jeweils diejenigen Matrixelemente, welche bis zum i–ten Schritt bei der Anwendung des jeweiligen Algorithmus bereits aufdatiert wurden.

Bemerkung 3.4.1. *Algorithmus I lässt sich hervorragend auf den Fall symmetrisch positiv definiter Matrizen A im Zusammenhang mit symmetrischen Rang-1 Aufdatierungen $A_+ = A + uu^T$ anwenden. Der Rechenaufwand, welcher zur Aufdatierung der dabei typischerweise verwendeten Cholesky- beziehungsweise LDL^T-Faktorisierung nötig ist, beträgt $2n^2$ Operationen. Dies ist signifikant weniger als der Rechenaufwand, welcher bei der Anwendung von Algorithmen, die auf orthogonalen Drehungen bzw. Spiegelungen beruhen [29], nötig ist.*

Dieses Vefahren lässt sich außerdem problemlos auf Aufdatierungsprobleme vom Rang

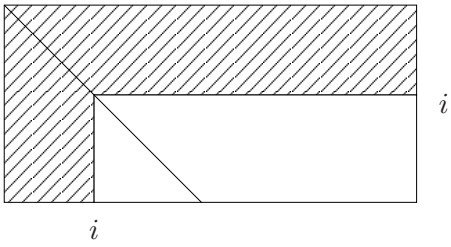


Abbildung 3.2: Standard Algorithmus

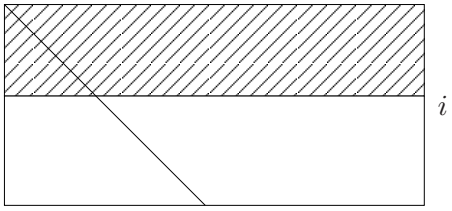


Abbildung 3.3: Zeilenweiser Algorithmus

$r > 1$ erweitern. Dabei bleibt das Vorgehen (3.7)-(3.10) nach intuitiver Anpassung der Dimensionen unverändert.

Bemerkung 3.4.2. *Der entscheidende Nachteil von Algorithmus I besteht darin, dass keine Permutationen der Dreiecksfaktorisierung möglich sind. Somit ergeben sich in einer Vielzahl praktischer Anwendungen numerische Stabilitätsprobleme.*

3.5 Algorithmus II

Dieser Algorithmus wurde von Kielbasinski und Schwetlick erstmals in [52] vorgestellt. Er ermöglicht die effiziente Rang-1 Aufdatierung der LU-Faktorisierung mit Zeilenpermutation. Im Weiteren ist zunächst das ursprüngliche Verfahren dargestellt. Darauf folgend werden einige neu entwickelte Erweiterungen vorgestellt. Im Gegensatz zu (3.5) wird mit Algorithmus II folgendes Problem behandelt

$$\underbrace{P_+^T L_+ U_+}_{A_+} = \underbrace{P^T LU}_{A} + uv^T \quad . \tag{3.11}$$

Durch Ausklammern in (3.11) erhält man in einem ersten Schritt

$$A^+ = P^T L(U + \tilde{u}v^T) \quad \text{mit} \quad P^T L\tilde{u} = u.$$

Dies bedeutet, dass das ursprüngliche Rang-1 Aufdatierungsproblem der Matrix A zunächst zu einem Rang-1 Problem die Matrix U betreffend umformuliert wird. Eine direkte Addition des Terms $\tilde{u}v^T$ zu U würde deren obere Dreiecksform zerstören. Deswegen wird zunächst durch eine wiederholte Anwendung von elementaren Transformationen T_i der Vektor $\tilde{u}$ auf ein Vielfaches des ersten Einheitsvektors reduziert. Während dieser Transformation werden die Elemete $\tilde{u}_i$ mit $i = n, \ldots, 2$ von unten beginnend eliminiert. Dies geschieht jeweils durch die Multiplikation eines Vielfachen des darüber liegenden Vektoreintrags. Um numerische Instabilitäten durch die Elimination betragsmäßig großer-

durch betragsmäßig kleine Elemente zu vermeiden, sind während dieses Prozesses Permutationen unabdingbar. Im einfachen Fall, das heisst falls $|\tilde{u}_i| \leq |\tilde{u}_{i-1}|$ ist, sind die Transformationen T_i durch elementare Dreiecksmatrizen gegeben, so dass im i–ten Schritt

$$A^{(i)} = P_{(i)}^T (L_{(i)} T_i^{-1})(T_i U_{(i)} + T_i \tilde{u}_{(i)} v^T)$$

die Dreiecksform von L erhalten bleibt. Hierbei wird T_i so gewählt, dass das Element $\tilde{u}_i$ eliminiert wird, und nur in der Matrix U ein zusätzliches Nicht-Null Element an der Position $(i, i-1)$ erzeugt wird. Dieses Nicht-Null Element wird im weiteren Verlauf des Algorithmus wieder eliminiert.

Im Fall, dass $|\tilde{u}_i| > |\tilde{u}_{i-1}|$ ist, setzt sich die Matrix T_i aus einem Produkt zweier elementarer Transformationen sowie einer Permutation zusammen

$$T_i = T_{i,L} T_{i,U} P_i.$$

Diese Matrizen sind, wie im Folgenden schematisch dargestellt ist, definiert.

Im Fall eines betragsmäßig kleinen Elements an der i-ten Stelle von $\tilde{u}$, hier durch ϵ dargestellt, sind Permutationen nötig, um $\tilde{u}$ numerisch stabil auf ein Vielfaches des ersten Einheitsvektors zu transformieren. Diese Transformationen $T_i = T_{i,L} T_{i,U} P_i$ können wie folgt definiert werden:

$$L(U + \tilde{u} v^T)$$

$$\equiv \left[\begin{array}{c}\diagdown\end{array}\right]\left(\left[\begin{array}{c}\diagdown\end{array}\right] + \begin{pmatrix}|\\ \epsilon \\ *\end{pmatrix}(* \text{—} *)\right)$$

a) P_i ist so gewählt, dass es die beiden Elemente $\tilde{u}_i$ und $\tilde{u}_{i+1}$ in $\tilde{u}$ vertauscht. Bei Anwendung dieser Permutation entsteht allerdings ein Nicht-Null Eintrag oberhalb der Diagonale in L_a.

$$P_i^T \underbrace{(P_i L P_i^T)}_{L_a}(\underbrace{P_i U}_{U_a} + \underbrace{P_i \tilde{u}}_{u_a} v^T)$$

$$\equiv P_i^T \left[\begin{array}{c}\diagdown \\ *\end{array}\right]\left(\left[\begin{array}{c}\diagdown \\ *\end{array}\right] + \begin{pmatrix}|\\ * \\ \epsilon\end{pmatrix}(* \text{—} *)\right)$$

b) $T_{i,U}$ ist die Einheitsmatrix mit einem zusätzlichen Eintrag oberhalb der Diagonale, welcher so gewählt wird, dass bei der Multiplikation $L_a T_{i,U}^{-1}$ das zuvor neu entstandene Element in L_a eliminiert wird.

$$P_i^T \underbrace{(L_a T_{i,U}^{-1})}_{L_b}(\underbrace{T_{i,U} U_a}_{U_b} + \underbrace{T_{i,U} u_a}_{u_b} v^T)$$

$$\equiv P_i^T \left[\begin{array}{c}\diagdown \\ 0\end{array}\right]\left(\left[\begin{array}{c}\diagdown \\ *\end{array}\right] + \begin{pmatrix}|\\ * \\ \epsilon\end{pmatrix}(* \text{—} *)\right)$$

c) $T_{i,L}$ ist eine Einheitsmatrix mit einem zusätzlichen Eintrag unterhalb der Diagonale, so dass bei der Multiplikation $T_{i,L}u_b$ die $(i+1)$–te Komponente von u_b zu Null wird.

$$P_i^T \underbrace{(L_b T_{i,L}^{-1})}_{L_0}\underbrace{(T_{i,L}U_b}_{U_0} + \underbrace{T_{i,L}u_b}_{u_0} v^T)$$

$$\equiv P_i^T \left[\begin{smallmatrix} & & \\ & & \\ & & \\ 0 & & \end{smallmatrix}\right] \left(\left[\begin{smallmatrix} & & * \\ & & \\ * & & \end{smallmatrix}\right] + \left(\begin{smallmatrix} | \\ * \\ 0 \end{smallmatrix}\right) (* \text{—} *) \right)$$

Die wiederholte Anwendung dieser Transformationen führt schließlich zu

$$\tilde{A} = \underbrace{(P^T P_u^T)}_{\tilde{P}^T}\underbrace{(P_u L T_u^{-1})}_{\tilde{L}}\underbrace{(T_u U}_{\tilde{U}} + \underbrace{T_u \tilde{u}}_{\hat{u}} v^T) \tag{3.12}$$

wobei

$$T_u = T_{m-1}T_{m-2}\cdots T_1, \quad P_u = P_{m-1}P_{m-2}\cdots P_1. \tag{3.13}$$

Hierbei ist $\tilde{U}$ eine obere Hessenberg-, $\tilde{P}$ eine Permutations- und $\tilde{L}$ eine untere Einsdreiecksmatrix.

Da $\hat{u}$ ein Vielfaches des ersten Einheitsvektors ist, kann jetzt der Niedrigrangterm $\hat{u}v^T$ zu $\tilde{U}$ addiert werden, ohne deren Hessenbergstruktur zu verändern. Damit erhält man

$$A_+ = \tilde{P}^T \tilde{L} \hat{U}.$$

In einem letzten Schritt werden jetzt die Subdiagonalelemente von $\hat{U}$ eliminiert, was erneut durch eine Folge von Transformationen der Art (3.13) möglich ist. Hierbei sind die Matrizen T_i so gewählt, dass die zusätzlichen Subdiagonaleinträge in $\hat{U}$ von oben beginnend eliminiert werden, das heißt

$$A_+ = \underbrace{(\tilde{P}^T P_d^T)}_{P_+}\underbrace{(P_d \tilde{L} T_d^{-1})}_{L_+}\underbrace{(T_d \tilde{U})}_{U_+} \quad , \tag{3.14}$$

wobei

$$T_d = T_1' T_2' \cdots T_{m-1}', \quad P_d' = P_1' P_2' \cdots P_{m-1}' \tag{3.15}$$

sind. Nach der Multiplikation (3.14) ergibt sich schließlich die LU-Zerlegung (3.4) der Matrix A_+ mit den aufdatierten Faktoren P_+, L_+ und U_+.

Die Durchführung dieses Algorithmus erfordert $kmn + \mathcal{O}(m) + \mathcal{O}(n)$ Operationen, wobei $5 \leq k \leq 9$. Dabei wird für k der kleinste Wert angenommen, wenn keine Permutationen nötig sind und der größte Wert, falls in jedem Eliminationsschritt permutiert werden muss.

Bemerkung 3.5.1. *Die Bedingung $|u_i| > |u_{i-1}|$ führt auf eine große Anzahl von Permutationen, wodurch der Algorithmus deutlich ausgebremst wird. Durch die Einführung eines Parameters $0 < \tau < 1$ und der neuen Permutationsbedingung $\tau|u_i| > |u_{i-1}|$ ist es möglich, die Anzahl der Zeilenvertauschungen zu reduzieren. Dabei ist allerdings zu beachten, dass mit kleinerem τ die Stabilitätseigenschaften des Verfahrens nachlassen.*

Bemerkung 3.5.2. *Auch bei Algorithmus II erfolgt eine spaltenweise Aufdatierung der Matrix L, vgl. Kapitel 3.4. Die Umformulierung zu einem zeilenweisen Vorgehen ist nur möglich, wenn $\tau = 0$ gewählt wird und damit keine Permutationen mehr möglich sind. In diesem Fall müssen während der Transformation von $\tilde{u}$ und $\hat{U}$ die Matrizen T_u bzw. T_d zunächst zwischengespeichert werden. Die zeilenweise Aufdatierung von L ist dann im Nachhinein durch die entsprechenden Multiplikationen $\tilde{L} = LT_u$ sowie $L_+ = \tilde{L}T_d$ möglich. In diesem Fall reduziert sich der Gesamtrechenaufwand zu $5nm$ Operationen.*

3.6 Aufdatierung mit Spaltenpermutationen

Mit keinem der beiden bisher beschriebenen Verfahren ist es möglich, eine LU-Zerlegung mit Spaltenpermutation aufzudatieren. Speziell im Fall von rechteckigen Matrizen, d.h. $m < n$, sind Spaltenpermutationen zur Aufrechterhaltung der Nichtsingularität des vorderen, quadratischen Blocks von U allerdings unabdingbar. Im Folgenden wird eine Verbesserung des in [64] vorgeschlagenen Algorithmus zur Spaltenpermutation vorgestellt.

Ein Austausch von Spalten ist genau dann nötig, wenn wie in (3.16) dargestellt, nach erfolgter Aufdatierung ein Hauptdiagonalelement der Matrix U_+ betragsmäßig sehr klein oder gleich Null ist. Das heißt, es ist

$$U^{(1)} = \begin{array}{c} \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & u_i & & & \end{array} \\ \left[\begin{array}{ccccc|cc} \times & \times & \times & \times & | & \times & \times \\ 0 & \epsilon & \times & \times & | & \times & \times \\ 0 & 0 & \times & \times & | & \times & \times \\ 0 & 0 & 0 & \times & | & \times & \times \end{array}\right] \end{array} \equiv \Big[U_1 \quad | \quad U_2 \Big] \quad , \tag{3.16}$$

mit $0 \leq |\epsilon| \ll \max\limits_{1 \leq i \leq m} (|u_{ii}|)$.
In diesem Fall muss die Spalte u_i mit einer geeigneten Spalte u_j aus dem hinteren Block U_2 getauscht werden. Dieser Vorgang entspricht der Spaltenpermutation

$$U^+ = UQ_{ij},$$

wobei $1 \leq i \leq m < j \leq n$ und Q_{ij} die Permutationsmatrix ist, welche genau die Spalten i und j tauscht. Das im Folgenden vorgestellte Verfahren ermöglicht es, einen solchen Spaltentausch auf eine Rang-1 Modifikation der Matrix A zurückzuführen.

Um zunächst eine geeignete Spalte u_j zur Permutation zu bestimmen, wird im ersten Schritt die Reihenfolge der Spalten in U_1 gemäß

$$U^{(2)} = U^{(1)}Q^{(1)} = \begin{array}{c} \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & u_i & & & \end{array} \\ \left[\begin{array}{cccc|cc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \epsilon & \times & \times \\ 0 & \times & \times & 0 & \times & \times \\ 0 & 0 & \times & 0 & \times & \times \end{array}\right] \end{array} \tag{3.17}$$

vertauscht. Dies entspricht einer Spaltenpermutation $U^{(2)} = U^{(1)}Q^{(1)}$, nach welcher $U^{(2)}$ von oberer Hessenbergform ist.

Im nächsten Schritt wird durch eine Folge von Transformationen (3.15) der führende Block von $U^{(2)}$ wieder in obere Dreiecksgestalt gebracht:

$$U^{(3)} = TU^{(2)} = \begin{array}{c} \begin{array}{ccccccc} & & & \tilde{u}_i & & & u_j \end{array} \\ \left[\begin{array}{cccc|cc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \tilde{\epsilon} & \times & u_{mj} \end{array}\right] \end{array} . \tag{3.18}$$

Durch diese Multiplikation wurde ϵ an die Position (m, m) von $U^{(3)}$ gebracht. Aufgrund der Eigenschaften der Transformationen T gilt ebenfalls $0 \leq |\tilde{\epsilon}| \ll \max\limits_{1 \leq i < m} (|u_{ii}^{(3)}|)$.

Durch eine weitere Permutation wird letztendlich die Spalte $\tilde{u}_i$ mit einer geeigneten der hinteren Spalten u_j, $m < j \leq n$ ausgetauscht. Die naheliegende Wahl ist hierbei diejenige Spalte u_j, welche das betragsmäßig größte Element in der letzten Zeile besitzt, d.h. $|u_{m,j}| \leq |u_{i,j}|$ für $i = m, \ldots, n$ und $i \neq j$. Somit erhält man

$$U^{+} = U^{(3)}Q^{(2)} = \begin{array}{c} \begin{array}{ccccccc} & & & u_j & & & \tilde{u}_i \end{array} \\ \left[\begin{array}{cccc|cc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & u_{mj} & \times & \tilde{\epsilon} \end{array}\right] \end{array} . \tag{3.19}$$

Werden die Schritte (3.16)-(3.19) explizit auf U ausgeführt, muss die Inverse der Transformationsmatrix im Schritt (3.18) auch von rechts zu L multipliziert werden. Auf

diese Weise würde die Spaltenpermutation zu folgenden aufdatierten Matrizen führen:

$$\begin{aligned} L_+ &= LT^{-1} \\ U_+ &= TU^{(1)}Q^{(1)}Q^{(2)} \\ Q_+^T &= Q^TQ^{(1)}Q^{(2)}. \end{aligned}$$

Alternativ ist es möglich, das Vorgehen (3.16)-(3.19) lediglich zur Bestimmung der Spalte u_j, welche mit u_i ausgetauscht werden soll, durchzuführen. Die dabei auftretenden Transformationen werden hierbei nicht explizit auf U ausgeführt. Stattdessen wird die entsprechende Spaltenpermutation

$$U^+ = U + \Big(u_i - u_j\Big)\Big(e_j - e_i\Big)^T \tag{3.20}$$

mit der Rang-1 Modifikation

$$P_+^TL_+U_+ = P^TLU + P^TL^{-1}\Big(u_i - u_j\Big)\Big(e_j - e_i\Big)^T \tag{3.21}$$

der Matrix A durchgeführt. Die Matrix Q_+ ergibt sich in diesem Fall aus $Q_+^T = Q^TQ_{ij}$, wobei Q_{ij} genau die Permutationsmatrix ist, welche die Spalten i und j vertauscht.

Je nach Vorgehensweise sind zum Austausch zweier Spalten weitere $knm + \mathcal{O}(n) + \mathcal{O}(m)$ Operationen, mit $5 \leq k \leq 9$ nötig. Da aufgrund einer Rang-1 Aufdatierung maximal eine Spaltenpermutation nötig ist, beschränkt sich der Gesamtaufwand zur Rang-1 Aufdatierung, inklusive Spaltenpermutation, demnach auf schlechtestenfalls $18mn + \mathcal{O}(n) + \mathcal{O}(m)$ Operationen.

3.7 Algorithmus III

In diesem Abschnitt wird ein neu entwickelter hybrider Algorithmus [67] vorgestellt, welcher zum einen die Verfahren aus (3.3) und (3.5) kombiniert und zum anderen um die Möglichkeit von Spaltenpermutationen erweitert. Weiterhin wird der Spezialfall ausgenutzt, in welchem einer der Vektoren u bzw. v mit einem führenden Nullblock beginnt. Letzteres ist zum Beispiel bei der Anwendung im Optimierungspaket [36] im Fall von linearen Nebenbedingungen von Bedeutung.

Das im Folgenden vorgestellte Aufdatierungsverfahren besteht aus drei Schritten:

(1) Ausnutzen von führenden Nullen in den Aufdatierungsvektoren und eine daraus folgende Reduktion der Dimension des Problems.

(2) Aufdatierung mit Algorithmus I, solange dies stabil möglich ist.

(3) Aufdatierung mit Algorithmus II, erweitert um Spaltenpermutationen, sobald Permutationen nötig sind.

Im Detail stellen sich die Schritte (1)-(3) wie folgt dar. Zur Vereinfachung der Schreibweise soll hierbei angenommen werden, dass $P = I$ ist.

Schritt 1: Beginnt einer der Vektoren u bzw. v mit einem Block führender Nullen, reduziert sich das Aufdatierungsproblem. Dies wird am Beispiel $u^T = \begin{bmatrix} 0 & \cdots & 0 & u_2^T \end{bmatrix}$ illustriert. In Blockschreibweise mit zu $u_2 \in \mathbb{R}^{(m-l)}$ passenden Dimensionen ergibt sich die Darstellung

$$\begin{aligned} A^+ &= \begin{bmatrix} L_{11}^+ & 0 \\ L_{21}^+ & L_{22}^+ \end{bmatrix} \begin{bmatrix} U_{11}^+ & U_{12}^+ \\ 0 & U_{22}^+ \end{bmatrix} \\ &= \begin{bmatrix} L_{11}^+ U_{11}^+ & L_{11}^+ U_{12}^+ \\ L_{21}^+ U_{11}^+ & L_{21}^+ U_{12}^+ + L_{22}^+ U_{22} \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} + u_2 v_1^T & L_{21}U_{12} + L_{22}U_{22} + u_2 v_2^T \end{bmatrix}}_{A+uv^T}. \end{aligned}$$

Aufgrund des Nullblocks in u erhält man sofort $L_{11}^+ = L_{11}$, $U_{11}^+ = U_{11}$ sowie $U_{12}^+ = U_{12}$. Für die verbleibenden Matrizen ergibt sich zunächst aus

$$L_{21}^+ U_{11}^+ = L_{21}U_{11} + u_2 v_1^T,$$

dass L_{21}^+ wie folgt berechnet werden kann:

$$(L_{21}^+ - L_{21})U_{11} = u_2 v_1^T \Longleftrightarrow L_{21}^+ = L_{21} + u_2 v_1^T U_{11}^{-1}.$$

Weiterhin gilt für L_{22}^+ und U_{22}^+

$$\begin{aligned} L_{21}^+ U_{12} + L_{22}^+ U_{22}^+ &= L_{21}U_{12} + L_{22}U_{22} + u_2 v_2^T \Longleftrightarrow \\ (L_{21}^+ - L_{21})U_{12} + L_{22}^+ U_{22}^+ &= L_{22}U_{22} + u_2 v_2^T \Longleftrightarrow \\ L_{22}^+ U_{22}^+ &= L_{22}U_{22} + u_2 \underbrace{(v_2^T - v_1^T U_{11}^{-1} U_{12})}_{\tilde{v}^T}. \end{aligned} \tag{3.22}$$

Aus (3.22) folgt, dass die Aufdatierung von L_{22} und U_{22} der auf die Dimension $(m-l) \times (n-l)$ reduzierten Rang-1 Modifikation einer LU-Zerlegung entspricht.

Schritt 2: Das in der Dimension reduzierte Problem (3.22) wird nun zunächst mit der zeilenweisen Variante des Algorithmus I aufdatiert. Dies geschieht so lange, wie keine Permutationen nötig sind. Als Übergangskriterium von diesem Verfahren zum dritten Schritt kann beispielsweise die Bedingung

$$|U_{ii}^{+}| \leq \kappa \cdot \max(|U_{i2}^{+}|) \quad \text{mit} \quad (0 \leq \kappa \leq 1)$$

verwendet werden. Des Weiteren muss noch der Block $\tilde{L}_{21}$, vgl. Abb. 3.4, berechnet werden. Dies geschieht durch die auf diese Teilmatrix eingeschränkte Verwendung der Standardversion des Algorithmus I.

Schritt 3: Für das verbleibende, erneut in der Dimension reduzierte Aufdatierungsproblem der Matrizen $\hat{L}$ und $\hat{U}$ wird nun Algorithmus II unter Hinzunahme von Spaltenpermutationen verwendet. Die in diesem Schritt auftretenden Permutationen müssen außerdem in den bereits modifizierten Matrizen $L_{21}^{+}, \tilde{L}_{21}^{+}, U_{12}^{+}, \tilde{U}_{12}^{+}$, vgl. Abb. 3.4, durchgeführt werden.

Eine schematische Darstellung des neuen hybriden Algorithmus zeigt Abb. 3.4. Der Vor-

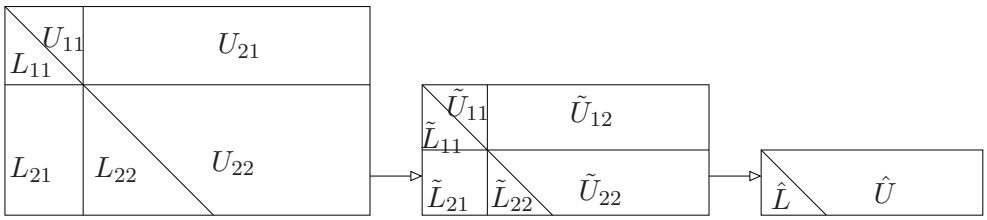


Abbildung 3.4: Schematische Darstellung des Algorithmus III

teil dieser Methode besteht darin, dass das ursprüngliche Problem zunächst mit sehr effizienten Verfahren in der Dimension reduziert wird. Der rechenintensivere Algorithmus muss dann nur noch auf das verbleibende, verkleinerte Problem angewendet werden. Wie stark diese Reduktion tatsächlich ausfällt, ist sowohl von der ursprünglichen LU Faktorisierung als auch vom zu addierenden Rang-1 Term abhängig.

Bemerkung 3.7.1. *In Schritt 1 ist ein analoges Vorgehen auch bei einem führenden Nullblock im Vektor v möglich, siehe dazu [64].*

3.8 Anwendung

Der im vorherigen Abschnitt vorgestellte Aufdatierungsalgorithmus wurde als Alternative zu einer auf QR-Faktorisierungen basierenden Variante des Optimierungspakets LRAMBO entwickelt. Diese Anwendung wird im Folgenden kurz dargestellt.

Gegeben sei das folgende beschränkte, nichtlineare Optimierungsproblem

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{wobei} \quad \begin{cases} c_i(x) = 0 & i \in \mathcal{E} \\ c_j(x) \leq 0 & j \in \mathcal{I} \end{cases} \quad \text{wobei } \mathcal{I} \cap \mathcal{E} = \emptyset, \ (\mathcal{I} \cup \mathcal{E}) = \{1, ..., k\},$$

mit den Gleichungsnebenbedingungen c_i sowie den Ungleichungsnebenbedingungen c_j.

Nach der Bestimmung von $m \leq k$ aktiven Bedingungen ist die Lösung dieses Optimierungsproblems durch die Sattelpunkte der Funktion

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x), \quad \text{mit } \lambda = (\lambda_1, ..., \lambda_m)^T$$

charakterisiert. Diese Sattelpunkte können durch die Lösung der Gleichung

$$0 = \nabla_{x,\lambda} \mathcal{L}(x, \lambda) \equiv [g(x, \lambda), c(x)],$$

mit $g(x, \lambda) = \nabla f + \sum_{i=1}^{m} \lambda_i \nabla c_i(x)$ bestimmt werden.

Dies ist beispielsweise mit Hilfe des quasi-Newton Verfahrens aus [35] möglich. Hierbei ist eine Folge von linearisierten KKT-Systemen [54] der Form

$$\begin{bmatrix} B_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} s_k \\ \sigma_k \end{bmatrix} = - \begin{bmatrix} g_k \\ c_k \end{bmatrix} \tag{3.23}$$

zu lösen. In (3.23) approximiert $A_k \in \mathbb{R}^{m\times n}$ die Jacobimatrix $\nabla c(x)^T$ der aktiven Nebenbedingungen und $B_k \in \mathbb{R}^{n\times n}$ die Hessematrix der Lagrangefunktion $\nabla_x^2 \mathcal{L} = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x)$. Die Vektoren $s_k \in \mathbb{R}^n$ und $\sigma_k \in \mathbb{R}^m$ sind die aktuellen Schrittweiten zur Berechnung der neuen Iterierten der Lösung

$$x_{k+1} = x_k + \alpha_k s_k$$

beziehungsweise der Langrange Multiplikatoren

$$\lambda_{k+1} = \lambda_k + \alpha_k \sigma_k.$$

Der Skalar α_k ist hierbei ein Dämpfungsparameter, welcher durch Liniensuche [9] bestimmt werden kann. Durch diesen wird die globale Konvergenz des Verfahrens sichergestellt.

Aufgrund der Verwendung eines quasi-Newton Verfahrens werden die approximierte Jacobimatrix A_k sowie die approximierte Hessematrix B_k in jedem Iterationsschritt k aufdatiert. Unter Zuhilfenahme des Automatischen Differenzierens [33], welches die effiziente Multiplikation von Vektoren mit Ableitungsmatrizen ermöglicht, ergeben sich für A_k und B_k die folgenden Rang-1 Modifikationen.

3.8.1 Aufdatierung der Ableitungsmatrizen

Die Hessematrix B_k wird in jedem Schritt durch die symmetrische Rang-1 Aufdatierungsformel (SR1) berechnet [15]:

$$B_{k+1} = B_k + \frac{(w_k - B_k s_k)(w_k - B_k s_k)^T}{(w_k - B_k s_k)^T s_k} \equiv B_k + \epsilon_k h h_k^T, \tag{3.24}$$

wobei durch

$$w_k = B_k s_k \equiv g(x_{k+1}, \lambda_k) - g(x_k, \lambda_k) \tag{3.25}$$

die direkte Sekantenbedingung erfüllt ist. Die rechte Seite von (3.25) kann mit dem adjungierten Modus des Automatischen Differenzierens ohne explizite Kenntnis von $\nabla c(x)$ berechnet werden.

Die Jacobimatrix A_k wird ähnlich zu (3.24) mit Hilfe einer beidseitigen Aufdatierungsformel (TR1) aktualisiert [35]:

$$A_{k+1} = A_k + \frac{(y_k - A_k s_k)(\mu_k^T - \sigma_k^T A_k)}{\mu_k^T s_k - \sigma_k^T A_k s_k} \equiv A_k + \delta_k r_k \rho_k^T. \tag{3.26}$$

Gleichung (3.26) erfüllt die direkte Sekantenbedingung

$$y_k = A_k s_k \equiv c(x_{k+1}) - c(x_k) \tag{3.27}$$

sowie die adjungierte Sekantenbedingung

$$\sigma_k^T A_{k+1} = \mu_k^T \equiv \sigma_k^T \nabla c(x_{k+1}) \tag{3.28}$$

bis zu $\mathcal{O}(\|\sigma_k\|\|s_k\|^2)$.
Auch in diesem Fall können (3.27) und (3.28) durch den rückwärts- bzw. adjungierten Modus des Automatischen Differenzierens berechnet werden. Weitere Eigenschaften der SR1- und TR1-Aufdatierungsformeln sind unter anderem in [35, 50] gegeben.

3.8.2 Nullraumdarstellung

Zur Berechnung der aktuellen Schrittweiten s_k und σ_k muss das lineare Gleichungssystem (3.23) gelöst werden. Dazu ist eine Zerlegung der Matrizen $A_k \in \mathbb{R}^{m\times n}$ sowie $B_k \in \mathbb{R}^{n\times n}$ nötig. Im Gegensatz zur QR-Zerlegung in [36] soll A_k an dieser Stelle durch die LU-Faktorisierung mit vollständiger Permutation zerlegt werden, d.h. es ist

$$P_k A_k Q_k = L_k U_k \tag{3.29}$$

mit $U_k = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$.

Analog zum Vorgehen in [36] wird die approximierte Hessematrix bezüglich des Nullraums von A_k sowie des Bildraums von A_k^T transformiert. Dabei wird

$$Z_k = Q_k \tilde{Z}_k \quad \text{mit} \quad \tilde{Z}_k = \begin{bmatrix} -U_1^{-1}U_2 \\ I_d \end{bmatrix} = \begin{bmatrix} \hat{Z} \\ I_d \end{bmatrix} \in \mathbb{R}^{n\times d}, d = n - m \tag{3.30}$$

als Basis für den Nullraum, sowie

$$Y_k = Q_k \tilde{Y}_k = Q_k \begin{bmatrix} I_m \\ 0 \end{bmatrix} \in \mathbb{R}^{n\times m} \tag{3.31}$$

als Basis für den Bildraum von A_k^T verwendet.

Durch die Transformation von (3.23) mit (3.30) und (3.31) ergibt sich folgende Darstellung:

$$\begin{bmatrix} E_k & C_k & U_1^T L_k^T P_k \\ C_k^T & M_k & 0 \\ P_k^T L_k U_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_y \\ s_z \\ \sigma_k \end{bmatrix} = - \begin{bmatrix} Y_k^T g_k \\ Z_k^T g_k \\ c_k \end{bmatrix}, \tag{3.32}$$

mit

$$\begin{aligned} E_k &= Y_k^T B_k Y_k \in \mathbb{R}^{m\times m} \\ C_k &= Y_k^T B_k Z_k \in \mathbb{R}^{m\times d} \\ M_k &= Z_k^T B_k Z_k \in \mathbb{R}^{d\times d} \end{aligned}$$

und $s_k = Y_k^T s_y + Z_k^T s_z$.

Zur Lösung von (3.32) muss weiterhin die Matrix M zerlegt werden. Aus Implementationsgründen, siehe [64], wird dazu eine transponierte Choleskyfaktorisierung $M_k = R_k R_k^T$ verwendet. Hierbei ist R_k eine obere Dreiecksmatrix. Das System wird in (3.24) und (3.26) gezielt gedämpft, so dass die Folge der Matrizen $(B_k)_k$ positiv definit bleibt. Als Startnäherungen bieten sich $A_0 = \begin{bmatrix} I_m & 0 \end{bmatrix}$ sowie $B_0 = I_n$ an.

Bemerkung 3.8.1. *Durch die Verwendung von (3.32) müssen sowohl Y_k, Z_k als auch B_k nicht explizit gespeichert werden. Bei der Anwendung dieses Verfahrens auf große, dichtbesetzte Systeme ermöglicht dies eine nicht zu vernachlässigende Speicherplatzersparnis.*

3.8.3 Aufdatierung der transformierten Hessematrix

Da eine direkte Berechnung der Faktorisierung (3.32), ausgehend von den explizit gegebenen Matrizen A_k und B_k im allgemeinen Fall einen Rechenaufwand der Größenordnung $\mathcal{O}(n^3)$ benötigt, soll dies möglichst vermieden werden. Aufgrund der initialen Wahl von A_0 und B_0 als Einheitsmatrizen ist die Ausgangszerlegung bereits trivial gegeben. Auch die weiteren Iterierten A_k und B_k müssen nicht explizit gespeichert werden. Da die SR1- sowie die TR1-Formel jeweils nur Rang-1 Korrekturen darstellen, lassen sich sämtliche in (3.32) auftretende Matrizen direkt und effizient aufdatieren. Dies ist in einer Komplexität von $\mathcal{O}(n^2)$ möglich. In den vorherigen Abschnitten wurde bereits dargestellt, wie hierfür die LU-Zerlegung aufdatiert werden kann. Die daraus resultierenden Aufdatierungen von Y_k und Z_k sowie der Matrizen E_k, C_k und R_k werden im Folgenden beschrieben.

(i) Aufdatierung der Basis des Nullraums Z_k sowie des Bildraums Y_k.

Durch die von der TR1-Aufdatierung hervorgerufene Rang-1 Korrektur von A_k ändert sich dessen Nullraum Z_k ebenfalls vom Rang eins. Es sei zunächst angenommen, dass zur Aufdatierung von A_k keine Spaltenpermutation nötig ist. In diesem Fall findet die Rang-1 Korrektur nur im oberen Block $\hat{Z}_k = -U_1^{-1}U_2$ von Z_k statt, vgl. (3.30).

Für die weitere Aufdatierung der transformierten Matrizen bezüglich des Nullraums Z_k ist eine explizite Darstellung dieses Niedrigrangterms notwendig. Gesucht sind demzufolge die Vektoren $z = \begin{bmatrix} \hat{z} \\ 0 \end{bmatrix} \in \mathbb{R}^n$ sowie $\rho_z \in \mathbb{R}^d$, so dass

$$\tilde{Z}_{k+1} = \tilde{Z}_k + z\rho_z^T, \tag{3.33}$$

und $\tilde{Z}_{k+1}$ eine Basis des Nullraums von A_{k+1} bildet. Diese Aufdatierungsvektoren lassen sich mit Hilfe der Sherman-Morrison-Woodbury Formel, siehe Kapitel 2.4, wie folgt berechnen.

Ausgehend von $A_{k+1} = A_k + \delta_k r_k \rho_k^T$ ergibt sich mit $\tilde{r}_k = \delta_k L_k^{-1} P_k^T r$ und $\begin{bmatrix} \rho_1^T & \rho_2^T \end{bmatrix} =$

$\rho_k^T Q_k$:

$$\begin{aligned} \hat{Z}_{k+1} &= -[U_1 + \tilde{r}_k \rho_1^T]^{-1}[U_2 + \tilde{r}_k \rho_2^T] && (3.34) \\ &= \hat{Z} \underbrace{-U_1^{-1}\tilde{r}_k}_{\hat{z}} \underbrace{\left[\rho_2^T - \frac{\rho_1^T U_1^{-1}(U_2 + \tilde{r}_k \rho_2^T)}{1 + \rho_1^T U_1^{-1} \tilde{r}_k}\right]}_{\rho_z^T} && (3.35) \\ &= \hat{Z} + \hat{z}\rho_z^T. && (3.36) \end{aligned}$$

Ist bei der Aufdatierung der LU-Faktorisierung von A_k eine Spaltenpermutation nötig, bedingt dies weitere Korrekturen in $Z_{k+1} = Q_{k+1}\tilde{Z}_{k+1}$. Zusätzlich zu (3.36) ist in diesem Fall eine weitere Rang-1 Modifikation, hervorgerufen durch den Spaltentausch in U_k nötig. Diese lässt sich, wie in Kapitel 3.6 beschrieben, darstellen. Die Matrix Q_k kann daraufhin in der aufdatierten Nullraumbasis durch Q_{k+1} ersetzt werden. Beide Rang-1 Modifikationen können für die folgenden Aufdatierungen der projizierten Matrizen zu einem Rang-2 Term zusammengefasst werden.

Eine explizite Aufdatierung der Bildraumbasis Y_k ist nicht nötig, da Y_{k+1} nach Definition (3.31) bereits durch Q_{k+1} eindeutig beschrieben ist.

(ii) Aufdatierung der transformierten Hessematrix.

Die Matrizen E_k, C_k sowie M_k unterliegen ebenfalls einer Niedrigrangaufdatierung in jedem Schritt des quasi-Newton Verfahrens. Diese Änderungen erfolgen aufgrund der SR1-Aufdatierung von B_k sowie der TR1-Aufdatierung von A_k und den damit verbundenen Änderungen der Nullraumbasis Z_k sowie der Bildraumbasis Y_k. Hierbei ist zu beachten, dass keine der Matrizen Y_k, Z_k und B_k gespeichert wird und diese somit nicht explizit vorliegen. Die aufdatierten Matrizen lassen sich im Einzelnen wie folgt berechnen:

- Die Matrix E_k ändert sich auf Grund der Rang-1 Korrektur von B_k wie folgt:

$$\begin{aligned} E_{k+1} &= Y_k^T B_{k+1} Y_k \\ &= Y_k^T (B_k + \epsilon_k h_k h_k^T) Y_k \\ &= \underbrace{Y_k^T B_k Y_k}_{E_k} + \epsilon_k Y_k^T h_k h_k^T Y_k \quad . && (3.37) \end{aligned}$$

Für eine auftretende Spaltenpermutation $Q_{k+1} = Q_k Q_{ij}$, bei welcher die i-te mit der j-ten Spalte getauscht wird, ist

$$\begin{aligned} E_{k+1} &= Y_{k+1}^T B_k Y_{k+1} \\ &= Y_k^T Q_{ij}^T B_k Q_{ij} Y_k. \end{aligned}$$

In diesem Fall ist es am effizientesten die i-te Zeile e_z beziehungsweise Spalte e_s von E_k mit denen durch

$$e_{z+1} = e_j^T Q_{k+1}^T B_k Y_{k+1} \text{ und } e_{s+1} = z_{k+1}^T \tag{3.38}$$

explizit berechneten zu ersetzen. Da die Matrix B_k nicht explizit vorliegt, muss diese in (3.38) durch ihre faktorisierte Darstellung ersetzt werden:

$$\begin{bmatrix} Y_k^T \\ Z_k^T \end{bmatrix} B_k \begin{bmatrix} Y_k & Z_k \end{bmatrix} = \begin{bmatrix} E_k & C_k \\ C_k^T & R_k R_k^T \end{bmatrix}$$
$$\Longrightarrow \quad B_k = Q_k \begin{bmatrix} I_m & 0 \\ -\hat{Z}_k^T & I_d \end{bmatrix} \begin{bmatrix} E_k & C_k \\ C_k^T & R_k R_k^T \end{bmatrix} \begin{bmatrix} I_m & -\hat{Z}_k \\ 0 & I_d \end{bmatrix} Q_k^T.$$

- Die Matrix $C_k = Y_k^T B_k Z_k$ unterliegt jeweils einer Niedrigrangmodifikation durch die Änderung der Matrizen Y_k, Z_k sowie B_k.

  Aufgrund der Modifikation von B_k berechnet sich C_{k+1} wie folgt:

$$\begin{aligned} C_{k+1} &= Y_k^T B_{k+1} Z_k \\ &= Y_k^T (B_k + \epsilon_k h_k h_k^T) Z_k \\ &= \underbrace{Y_k^T B_k Z_k}_{C_k} + \epsilon_k Y_k^T h_k h_k^T Z_k. \end{aligned} \tag{3.39}$$

  Im Fall einer Änderung des Nullraums ergibt sich durch analoges Vorgehen

$$\begin{aligned} C_{k+1} &= Y_k^T B_k Z_{k+1} \\ &= Y_k^T B_k Q_k (\tilde{Z}_k + \tilde{z} \rho_z^T) \\ &= \underbrace{Y_k^T B_k Z_k}_{C_k} + Y_k^T B_k Q \tilde{z} \rho_z^T. \end{aligned} \tag{3.40}$$

  Sollte eine Spaltenpermutation auftreten, so ist

$$C_{k+1} = Y_k^T Q_{ij}^T B_k Q_{ij} Z_k.$$

  Die Berechnung von C_{k+1} ist in diesem Fall in zwei Schritten möglich. Zunächst kann die Korrektur bezüglich Z_{k+1} wie in (3.40) durchgeführt werden. Im zweiten Aufdatierungsschritt (bezüglich Y_{k+1}) wird die i-te Zeile von C_k durch die analog zu (3.38) explizit berechnete, neue Zeile $c_{z+1}^T = e_j^T Q_{k+1} B_k Z_{k+1}$ ersetzt.

- Die Aufdatierung von $M_k = R_k R_k^T$ ist nötig, sobald sich B_k oder Z_k ändern. Im ersten Fall ist

$$\begin{aligned} M_{k+1} = R_{k+1} R_{k+1}^T &= Z_k^T B_{k+1} Z_k \\ &= Z_{k+1}^T (B_k + \epsilon_k h_k h - k^T) Z_k \\ &= \underbrace{R_k R_k^T}_{M_k} + \epsilon_k Z_k^T h_k h_k^T Z_k. \end{aligned} \tag{3.41}$$

  Diese Rang-1 Aufdatierung der transponierten Choleskyzerlegung kann zum Beispiel mit Algorithmus I oder einem auf orthogonalen Transformationen basierenden Verfahren [29] durchgeführt werden.

  Durch die Niedrigrangkorrektur der Nullraumbasis Z_k, welche auch im Fall einer Spaltenpermutation auftritt, ergibt sich für M_k folgende Rang-2 Aufdatierung:

$$\begin{aligned} M_{k+1} = R_{k+1} R_{k+1}^T &= (Z_k + z\rho_z^T)^T B_k (Z_k + z\rho_z^T) \\ &= \underbrace{Z_k^T B_k Z_k}_{M_k} + \rho_z b_z^T + b_z \rho_z^T + \mu_z \rho_z \rho_z^T, \end{aligned}$$

  wobei $b_z = Z_k^T B_k z$ und $\mu_z = z^T B_k Z_k$. Diese kann als Folge von zwei Rang-1 Änderungen analog zur Aufdatierung von (3.41) betrachtet werden.

3.9 Numerische Ergebnisse

In diesem Kapitel werden die verschiedenen hier vorgestellten Aufdatierungsalgorithmen bezüglich ihrer Laufzeit miteinander verglichen. Die hier angegebenen Zeitmessungen entstanden bei der Nutzung einer Athlon-XP 2100+ CPU mit 256kB CPU Cache und 512MB Hauptspeicher, unter Verwendung des Betriebssystem Linux. Die Algorithmen wurden in C implementiert und mittels des gcc Compilers und der Option -o3 compiliert.

Zunächst werden die reinen LU Aufdatierungen bezüglich ihrer Laufzeit untereinander verglichen. Außerdem wird die Entwicklung der 2-Norm der Matrixfaktoren L und U für die verschiedenen Verfahren dargestellt.
Im zweiten Abschnitt werden die verschiedenen Methoden im Optimierungspaket LRAMBO angewendet. Hierbei kommen sowohl die Algorithmen I,II und III sowie die in Kapitel 3.8.3 vorgestellten Aufdatierungen zum Einsatz. Das spezielle Interesse liegt erneut im Laufzeitvergleich der Algorithmen untereinander sowie in ihrer Kompetitivität im Vergleich zur alternativ verwendeten QR-Faktorisierung von A_k.

3.9.1 Allgemeine Resultate

Im ersten Beispiel werden die ursprünglichen Aufdatierungsalgorithmen von [6] und [52] sowie die neu entwickelten Modifikationen in der Laufzeit verglichen. Hierbei werden, beginnend mit der Einheitsmatrix $A = \begin{bmatrix} I_m & 0 \end{bmatrix} \in \mathbb{R}^{m \times n}$, nacheinander 50 zufällig generierte Rang-1 Korrekturen durchgeführt. Da bei der Verwendung von zufällig generierten Rang-1 Aufdatierungen Spaltenpermutationen äußerst unwahrscheinlich sind, und da auch weiterhin jegliche Struktur in den Aufdatierungsvektoren fehlt (z.B. führende Nullblöcke), wird in diesem Beispiel auf eine Gegenüberstellung zum hybriden Algorithmus III verzichtet. Die Laufzeiten in Tabelle 3.5 sind in Sekunden angegeben. Die

Dimension	Alg.II, $\tau = 0$		Alg.II, $\tau = 1$		Alg.II, $\tau = 0.1$		Alg.I	
$\times 10^3$		r-w		# piv		# piv		r-w
3×3	49.1	28.2	76.5	274459	51.5	16799	28.7	13.6
1.5×6	30.4	23.8	45.7	133754	31.1	8230	19.3	14.0
0.75×1.2	25.0	22.8	37.2	63880	25.9	3951	16.4	14.0
6×6	434.2	112.6	635.0	563154	448.0	34573	269.7	55.5
3×1.2	177.1	95.6	268.0	275213	182.4	17370	118.1	56.5
1.5×2.4	116.4	99.4	165.4	132816	120.3	8235	75.7	58.9

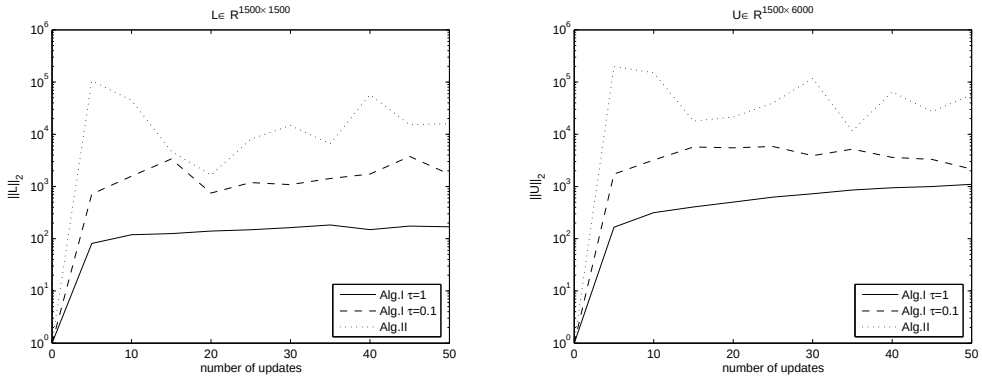


Abbildung 3.5: zufällig generierte Aufdatierung

Abkürzungen der Algorithmen bedeuten:

- Alg. I: Algorithmus I aus Kapitel 3.3, keine Permutation,

- Alg. I, r-w: Algorithmus I aus Kapitel 3.3, keine Permutation, zeilenweise Aufdatierung der Matrizen,
- Alg. II, $\tau = 0$: Algorithmus II aus Kapitel 3.5, keine Permutation,
- Alg. II, $\tau = 0$, r-w: Algorithmus II aus Kapitel 3.5, keine Permutation, zeilenweise Aufdatierung der Matrizen,
- Alg. II, $\tau = 1$: Algorithmus II aus Kapitel 3.5, mit Zeilenpermutation nach [52],
- Alg. II, $\tau = 0.1$: Algorithmus II aus Kapitel 3.5, mit reduzierter Zeilenpermutation.

In den mit '#piv' benannten Spalten ist jeweils die Gesamtanzahl der während der 50 Aufdatierungen durchgeführten Zeilenpermutationen angegeben.
In den Diagrammen der Abbildung 3.5 wird außerdem die Entwicklung der 2-Norm der Matrizen L und U, in Abhängigkeit der bereits durchgeführten Rang-1 Aufdatierungen, für die Algorithmen Alg.II ($\tau = 1$), Alg.II ($\tau = 0.1$) und Alg.I dargestellt.

Wie erwartet, ist die zeilenweise Version des Algorithmus I in jedem Fall am effizientesten. Besonders im Fall großer Dimensionen ist dieses Verfahren signifikant schneller als die anderen Methoden. Selbst die zeilenweise Implementierung von Algorithmus II, welche theoretisch betrachtet nur eine um 25% höhere Komplexität hat, benötigt 40% mehr Laufzeit. Dies ist dadurch begründet, dass in Algorithmus II jedes Matrixelement von L und U zweimal modifiziert werden muss. Diese zusätzlichen Speicherzugriffe erhöhen im Gegensatz zu Algorithmus I, in welchem jeder Matrixeintrag nur einmal geändert wird, die Laufzeit überproportional.

Bei der Verwendung des Algorithmus I ist es nicht möglich, durch Zeilen- bzw. Spaltenvertauschungen numerische Stabilität sicherzustellen. Aufgrund dieser fehlenden Permutationsmöglichkeit sind die Normen der Matrizen L und U deutlich größer als bei der Anwendung des Algorithmus II mit Permutationen. Im Vergleich der unterschiedlichen Wahl des Parameter τ stellt sich heraus, dass Algorithmus II mit $\tau = 0.1$ einen guten Kompromiss bezüglich Laufzeit und Stabilität bietet.

In einem zweiten Beispiel unterliegt die Einheitsmatrix erneut einer Sequenz von 50 zufällig erzeugten Rang-1 Modifikationen. Im Gegensatz zum ersten Test beginnt die Aufdatierungssequenz nun allerdings mit Vektoren großer Norm, welche Schritt für Schritt verkleinert wird. Dies simuliert die Struktur der Korrekturen, welcher die Matrix A_k während des quasi-Newton Verfahrens in Kapitel 3.8 unterliegt. Für diesen Fall werden die zeilenweise Version von Algorithmus I, Algorithmus II, der neu entwickelte hybride Algorithmus III sowie die Aufdatierung einer entsprechenden QR-Zerlegung, siehe [29],

miteinander verglichen. Die Bezeichnungen in Tabelle 3.6 entsprechen denen des ersten Beispiels. Bei Auswertung der Ergebnisse fällt zunächst auf, dass alle LU-basierten

Dimension	QR-basiert	LU-basiert		
$m \times n$		Alg.II	Alg.I	Alg.III
6000 × 6000	695.1	639.91	55.52	198.23
3000 × 6000	179.15	169.6	27.89	65.27
3000 × 3000	90.2	75.75	9.65	30.82
1500 × 3000	28.51	22.8	6.83	13.45

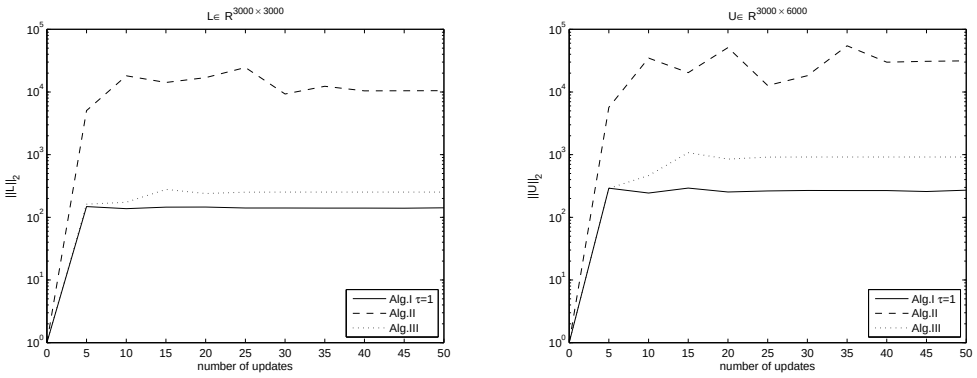


Abbildung 3.6: strukturierte Aufdatierung

Aufdatierungen schneller berechnet wurden als dies bei Nutzung der QR-Zerlegung der Fall war. Erneut liefert Algorithmus II die numerisch stabilsten Ergebnisse, wohingegen Algorithmus I die deutlich beste Laufzeit aufzuweisen hat. Als geeigneter Kompromiss bezüglich Laufzeit und Stabilität kristallisiert sich klar der neue Algorithmus III heraus. Dieser kann als einziger die spezielle Struktur der hier generierten Rang-1 Sequenz ausnutzen. Zu Beginn der Aufdatierungssequenz werden bei Anwendung dieses Verfahrens Permutationen durchgeführt, während gegen Ende fast die gesamte Rang-1 Korrektur im schnellen, zeilenweisen Modus durchgeführt werden kann. Daraus resultieren zum einen die niedrigen Normen der Matrizen L und U sowie die vergleichsweise kurzen Laufzeiten. Letztere würden sich im Fall von hier nicht auftretenden führenden Nullblöcken in den Aufdatierungsvektoren noch weiter reduzieren.

3.9.2 KKT-Anwendung

In diesem Abschnitt werden die verschiedenen Aufdatierungsalgorithmen bei der Lösung von KKT-Systemen verglichen. Auf der einen Seite wird ein QR-basierter Algorithmus, $A_k = Q_k R_k$, zusammen mit auf Givens Rotationen basierenden Aufdatierungen der Matrix B_k, auf der anderen Seite die LU-basierte Methode sowie die Aufdatierung der Projektionen von B_k wie in Kapitel 3.8.3 beschrieben, verglichen. Für die Aufdatierung der Dreieckszerlegung werden erneut die Algorithmen I,II und III verwendet. Als Optimierungsumgebung wird in allen Fällen [36] verwendet. Die Berechnung der rechten Seite von (3.23) erfolgt durch Automatisches Differenzieren [33].

Im ersten Beispiel wird folgendes Optimierungsproblem [48] betrachtet:

Minimiere

$$f(x) = \sum_{i=1}^{n-1} \left(x_i + x_{i+1}\right)^2, \tag{3.42}$$

so dass

$$c_i \equiv x_i + 2x_{i+1} + 3x_{i+2} - 1 = 0 \quad (1 \leq i \leq n-2). \tag{3.43}$$

Für den Startwert x werden zufällig generierte Einträge $-0.5 \leq x_i \leq 0.5$ verwendet. Als Initialisierungen der Ableitungsmatrizen wird $A_0 = \begin{bmatrix} I & 0 \end{bmatrix}$ sowie $B_0 = I$ in den passenden Dimensionen verwendet. Tabelle 3.1 zeigt die Laufzeiten, welche zur Lösung

Dimension	LU-basiert			QR-basiert
n	Alg.II,$\tau = 0.1$	Alg.I	Alg.III	
200	3.75	3.47	3.73	5.23
300	12.12	10.3	12.36	20.64
400	31.45	27.21	32.46	52.61
500	49.83	40.46	51.79	98.71

Tabelle 3.1: 1. KKT-Beispiel

von (3.42)–(3.43) in den entsprechenden Dimensionen nötig sind. Hierbei ist anzumerken, dass ungefähr 50% dieser Rechenzeit für Berechnungen, welche nicht direkt mit der Aufdatierung zusammenhängen, benötigt werden (z.B. Liniensuche, Automatisches Differenzieren). Dieser Aufwand ist nahezu identisch für alle der gegebenen Algorithmen. Aufgrund der Struktur dieses Problems sind so gut wie keine Permutationen während

des Aufdatierungsprozesses nötig. Deswegen konvergiert das Verfahren auch bei Nutzung von Algorithmus I und liefert dabei die beste Rechenzeit.

Im zweiten Beispiel soll folgendes Optimierungsproblem [48] gelöst werden.

Minimiere

$$f(x) = \sum_{i=1}^{m} x_{3i-2} + x_{3i-1} + x_{3i-2}x_{3i} + \frac{5}{2}x_{3i}, \tag{3.44}$$

so dass

$$c_i \equiv \frac{1}{3}x_{3i-2} - x_{3i} = 0 \quad (1 \leq i \leq m). \tag{3.45}$$

Als Startvektor wird $x_i = 1$ für $(1 \leq i \leq 3m = n)$ verwendet. Erneut sind die Anfangsnäherungen der Ableitungsmatrizen durch $A_0 = \begin{bmatrix} I & 0 \end{bmatrix}$ und $B_0 = I$ gegeben. Analog zum ersten Beispiel werden die verschiedenen LU-basierten Verfahren und ei-

Dimension	LU-basiert			QR-basiert
n	Alg.I,$\tau = 0.1$	Alg.II	Alg.III	
200	3.14	3.06	3.13	3.93
450	8.92	8.61	9.66	13.51
600	–	–	21.73	32.72

Tabelle 3.2: 2. KKT-Beispiel

ne QR Version in der Laufzeit verglichen. Ist keine Zeit angegeben bedeutet dies, dass das entsprechende Verfahren in diesem Fall nicht konvergiert. Aufgrund der stark rechteckigen Struktur der Matrix A_k, wodurch eine große Anzahl an Spaltenpermutationen bedingt wird, ist dies für Algorithmus I mit $\tau = 0.1$ und Algorithmus II bei großer Dimension der Fall. Da nur der neu entwickelte, hybride Algorithmus Spaltenpermutationen ermöglicht, ist dieser das einzige LU-basierte Verfahren, welches in jedem Fall zur Lösung führt. Erneut ist zu sehen, dass die QR-Variante für alle Dimensionen langsamer ist.

3.10 Zusammenfassung

In diesem Kapitel wurden zunächst zwei bekannte Rang-1 Aufdatierungsverfahren der LU-Faktorisierung vorgestellt. Diese ermöglichen die Berechnung der Matrixfaktoren der

aufdatierten Zerlegung in $4n^2$ Operationen für den Fall ohne Permutationen und in maximal $9n^2$ Operationen für den Fall mit Zeilenpermutation. Durch eine effiziente Implementierung konnte die Laufzeit des ersten Algorithmus durch optimierte Speicherzugriffe deutlich verringert werden. Des Weiteren wurde ein Algorithmus entwickelt, der basierend auf dem zweiten vorgestellten Aufdatierungsverfahren das Problem um Spaltenpermutationen erweitert, ohne die Größenordnung der zur Aufdatierung benötigten Operationen von $\mathcal{O}(n^2)$ zu erhöhen. Die Verwendung von Spaltenpermutationen sind im Allgemeinen für rechteckige Matrizen unverzichtbar.

Weiterhin wurde ein neuer hybrider Algorithmus vorgestellt, welcher zunächst das effizientere Verfahren nutzt und bei möglicher numerischen Instabilität zur Methode mit vollständiger Permutation wechselt. Dieser Algorithmus bietet einen guten Kompromiss aus Effizienz im Sinne von kurzer Laufzeit, und numerischer Stabilität.

Dieses neue Verfahren wurde speziell für die Anwendung im Optimierungspaket LRAMBO entwickelt. Bei der Lösung von nichlinearen, beschränkten Optimierungsproblemen mit diesem Paket treten weitere Niedrigrangmodifikationen auf. Es wurde dargestellt, wie diese ebenfalls mit einem Gesamtaufwand von $\mathcal{O}(n^2)$ Operationen berechnet werden können.

Sämtliche Resultate wurden im Kapitel 3.9 anhand von Beispielen belegt.

Da die vorgestellten Aufdatierungsverfahren größtenteils stark sequentiell ablaufen, wurde auf die Entwicklung eines parallelen Algorithmus verzichtet. Des Weiteren sind die neu entwickelten Methoden hauptsächlich zur Lösung großer, dichtbesetzter Probleme konzipiert. Somit wurde eine Optimierung in Bezug auf Dünnbesetztheitsstukturen in dieser Arbeit außen vor gelassen.

Kapitel 4

Aufdatierung der Singulärwertzerlegung

4.1 Motivation

Die Singulärwertzerlegung ist eine in zahlreichen Gebieten der Numerik verwendete Faktorisierung. Sie kann beispielsweise zur Lösung von Problemen der kleinsten Fehlerquadrate oder zur Bestimmung des Rangs einer Matrix verwendet werden. Damit ist die Singulärwertzerlegung eine wichtige Komponente zur Approximation von Matrizen. Sie findet in zahlreichen Gebieten, wie beispielsweise in der Signalverarbeitung [60], in Bildkompressionsverfahren [30], in der Teilchenphysik [56] oder in der Statistik ihre Anwendung [44]. Die Berechnung der Singulärwertzerlegung ist allerdings mit einem hohen Rechenaufwand von $\mathcal{O}(n^3)$ Operationen für eine Matrix der Dimension $(n \times n)$ verbunden.

Einige Anwendungen erfordern die Berechnung einer Folge von Singulärwertzerlegungen. Dabei werden typischerweise in jedem Schritt einzelne Zeilen oder Spalten an die ursprüngliche Matrix hinzugefügt bzw. von dieser entfernt. Im allgemeineren Fall unterliegt die gesamte Ausgangsmatrix einer Folge von Rang-1 Korrekturen, d.h. $A_{k+1} = A_k + uv^T$. Eine explizite Berechnung der aufdatierten Matrizen sowie die anschließende Neuberechnung ihrer Singulärwertzerlegung in jedem Schritt k würde einen extrem hohen numerischen Aufwand erzeugen.

In diesem Kapitel ist die Entwicklung eines Verfahrens von Interesse, welches es ermöglicht, die bereits bekannten Matrizen der Zerlegung der Ausgangsmatrix bezüglich der auftretenden Rang-1 Korrekturen direkt aufzudatieren. Es wird ein Algorithmus vor-

gestellt, welcher diese Aufdatierung in $\mathcal{O}(n^2 \log_2^2 n)$ Operationen ermöglicht. Für den Fall, dass die zugrundeliegende Matrix nicht von vollem Rang ist, wird dieser Aufwand weiter reduziert. Die Effizienz des im Folgenden dargestellten Algorithmus basiert hauptsächlich auf schnellen Matrix-Matrix Multiplikationen, welche durch spezielle Cauchy-Matrix Strukturen sowie Matrix Approximationtechniken erzielt werden können. Um eine hohe Genauigkeit der berechneten Lösung sicherzustellen, wird ein abschließender Korrekturschritt im Aufdatierungsalgorithmus beschrieben.

Das hier dargestellte Verfahren verfolgt einen ähnlichen Ansatz und stellt eine Erweiterung von [37] dar. Es wurde in [65] und [66] veröffentlicht.

4.2 Symmetrisches Eigenwertproblem

Gegeben sei die Eigenwertzerlegung

$$B = QDQ^T, \tag{4.1}$$

sowie das symmetrische Rang-1 Aufdatierungsproblem

$$B_+ = QDQ^T + \rho uu^T. \tag{4.2}$$

Hierbei ist die Matrix $Q \in \mathbb{R}^{n\times n}$ orthogonal und $D \in \mathbb{R}^{n\times n}$ diagonal. Die Diagonalelemente von $D = \operatorname{diag}(\lambda_i)_{i=1,\dots,n}$ sind die der Größe nach geordneten Eigenwerte $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ der Matrix B. Die Spalten von Q sind durch die Eigenvektoren der Matrix B definiert. Weiterhin ist durch den Skalar $\rho \in \mathbb{R}$ sowie den Vektor $u \in \mathbb{R}^n$ eine Rang-1 Modifikation für B gegeben.

Gesucht ist die analog zu (4.1) definierte Eigenwertzerlegung der aufdatierten Matrix

$$B_+ = Q_+ D_+ Q_+^T. \tag{4.3}$$

Mit Hilfe des Vorgehens aus [14] ist eine effiziente Berechnung der neuen Eigenwerte $\lambda_1^+ \leq \lambda_2^+ \leq \cdots \leq \lambda_n^+$, welche den Diagonalelementen der Matrix D_+ entsprechen wie folgt möglich.

Durch Ausklammern der Matrix Q lässt sich (4.2) zu

$$B = Q(D + \tilde{\rho}\tilde{u}\tilde{u}^T)Q^T \tag{4.4}$$

mit $\tilde{u} = \frac{Q^T u}{\|Q^T u\|_2}$ und $\tilde{\rho} = \rho\|u\|_2^2$ umschreiben. Mit

$$D_+ = D + \tilde{\rho}\tilde{u}\tilde{u}^T \tag{4.5}$$

gilt folgendes Theorem:

Theorem 4.2.1. *Seien D, D_+, $\tilde{\rho}$, $\tilde{u}$ sowie λ_i und λ_i^+ mit $i = 1, \ldots, n$ wie zuvor definiert. Dann ist*

$$\lambda_i^+ = \lambda_i + \tilde{\rho}\mu_i,$$

wobei $\sum_{i=1}^{n} \mu_i = 1$ und $0 \leq \mu_i \leq 1$. Weiterhin gilt, dass

$$\begin{aligned} \lambda_1 \leq \lambda_1^+ \leq \lambda_2 \leq \cdots \leq \lambda_n \leq \lambda_n^+, \quad & \textit{falls} \quad \tilde{\rho} > 0 \\ \lambda_1^+ \leq \lambda_1 \leq \lambda_2^+ \leq \cdots \leq \lambda_n^+ \leq \lambda_n, \quad & \textit{falls} \quad \tilde{\rho} < 0 \quad . \end{aligned} \tag{4.6}$$

Beweis. Siehe [70]. □

Mit diesen Vorüberlegungen erfolgt die Berechnung der Eigenwertzerlegung (4.4) in drei Schritten:

(1) Reduktion der Dimension des Systems.

(2) Berechnung der aufdatierten Eigenwerte durch das Lösen nichtlinearer Gleichungen.

(3) Aufdatierung der Eigenvektoren.

Die Punkte (1)-(3) werden in den nachfolgenden Kapiteln ausführlich behandelt.

4.2.1 Reduktion der Dimension

Wie bereits in (4.4) definiert, sei $\|\tilde{u}\|_2 = \| \begin{bmatrix} \tilde{u}_1 & \tilde{u}_2 & \cdots & \tilde{u}_n \end{bmatrix}^T \|_2 = 1$. In folgenden Situationen lässt sich das ursprüngliche Problem bezüglich der Dimension reduzieren:

(i) Im Fall $\tilde{u}_i = 0$ ist $\lambda_i^+ = \lambda_i$, da $\lambda_i^+ e_i = (D + \tilde{\rho}\tilde{u}\tilde{u}^T)e_i = De_i + \tilde{\rho}\tilde{u}\tilde{u}_i = \lambda_i e_i$.

(ii) Wenn für ein bestimmtes i gilt, dass $|\tilde{u}_i| = 1$ ist, so ist $\tilde{u}_j = 0$ für alle $j \neq i$. Damit ist $\lambda_i^+ = \lambda_i + \tilde{\rho}$ und $\lambda_j^+ = \lambda_j$ für alle $j \neq i$.

(iii) Wenn ein Eigenenwert λ_i die Vielfachheit $r > 1$ hat. Sei in diesem Fall $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ so partitioniert, dass Q_1 den zu λ_i gehörigen Unterraum aufspannt. D sei entsprechend permutiert. Damit ist zunächst

$$\tilde{u} = Q^T u = \begin{bmatrix} Q_1^T u \\ Q_2^T u \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad . \tag{4.7}$$

Weiterhin sei $H \in \mathbb{R}^{r\times r}$ genau die Spiegelungsmatrix, so dass $Hu_1 = -\sigma e_1$ und $\sigma = \|u_1\|_2$. Mit $\bar{Q}_1 = Q_1 H^T$ gilt somit

$$\begin{aligned} \bar{Q}_1^T u &= HQ_1^T u = Hu_1 = -\sigma e_1, \\ \bar{Q}_1^T \bar{Q}_1^T &= HQ_1^T Q_1 H^T = HH^T = I \\ \bar{Q}_1^T Q_2 &= HQ_1^T Q_2 = 0 \quad . \end{aligned} \tag{4.8}$$

Daraus folgt, dass sich Q_1 durch $\bar{Q}_1$ ersetzen lässt. Durch den Übergang von $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ zu $Q = \begin{bmatrix} \bar{Q}_1 & Q_2 \end{bmatrix}$ werden wegen Gleichung (4.8) $(r-1)$ Elemente des Vektors $\tilde{u}$ zu Null reduziert. Aufgrund von Punkt (i) bleiben damit $(r-1)$ Eigenwerte unverändert.

Nach den entsprechenden Permutationen in $\tilde{u}$, Q und D

$$\tilde{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ u_2 \end{bmatrix}, \quad Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}, \quad D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \tag{4.9}$$

mit $D_1, D_1^+ \in \mathbb{R}^{l\times l}$, $D_2, D_2^+ \in \mathbb{R}^{k\times k}$, $Q_1 \in \mathbb{R}^{n\times l}$, $Q_2 \in \mathbb{R}^{n\times k}$, $u_1 \in \mathbb{R}^l$, $u_2 \in \mathbb{R}^k$ und $k = n - l$, lässt sich (4.5) in folgender Blockmatrixnotation schreiben

$$\begin{aligned} D^+ &= D + \tilde{\rho}\tilde{u}\tilde{u}^T \\ \begin{bmatrix} D_1^+ & 0 \\ 0 & D_2^+ \end{bmatrix} &= \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} + \tilde{\rho} \begin{bmatrix} 0 & 0 \\ 0 & u_2 u_2^T \end{bmatrix}. \end{aligned} \tag{4.10}$$

Da somit die Eigenwerte der Matrix D_1 unverändert bleiben, reduziert sich das Aufdatierungsproblem (4.5) zu

$$D_2^+ = D_2 + \tilde{\rho} u_2 u_2^T \in \mathbb{R}^{k\times k}.$$

Hierbei besitzt die Matrix D_2 paarweise verschiedene Diagonalelemente. Sämtliche Einträge des Aufdatierungsvektors u_2 sind außerdem ungleich Null.

Bemerkung 4.2.1. *Um zu entscheiden ob einer der Fälle (i)-(iii) vorliegt, werden numerisch folgende Ungleichungen verwendet:*

- *Ein Eintrag des Vektors $\tilde{u}$ wird als Null definiert, wenn dieser betragsmäßig kleiner als eine Schranke ε ist, d.h. $|\tilde{u}_i| < \varepsilon \Rightarrow u_i := 0$.*
- *Zwei Eigenwerte werden dann als gleich angenommen, wenn die Ungleichung $|\lambda_{i+1} - \lambda_i| \leq \varepsilon|\lambda_1|$ gilt. In diesem Fall wird $\lambda_{i+1} := \lambda_i$ definiert. Hierbei bezeichnet λ_1 den betragsmäßig größten Eigenwert.*

Eine typische Wahl für ε ist $\varepsilon = n \cdot \epsilon$, wobei n die Dimension des Problems und ϵ die Maschinengenauigkeit ist.

4.2.2 Berechnung der aufdatierten Eigenwerte

Der Einfachheit halber soll im Folgenden von einem bereits reduzierten Aufdatierungsproblem mit der Notation aus (4.2) bzw. (4.4) sowie der Dimension n ausgegangen werden. In diesem Fall sind nach [26] die Eigenwerte von B_+ durch die Nullstellen der nichtlinearen Gleichung

$$w(\lambda) = 1 + \tilde{\rho} \sum_{j=1}^{n} \frac{\tilde{u}_j^2}{\lambda_j - \lambda} \tag{4.11}$$

gegeben. (4.11) wird auch als Säkulargleichung bezeichnet. Zur numerischen Berechnung der Nullstellen λ_i^+ von (4.11) eignet sich folgendes Vorgehen, vgl. [14]:

(i) Aufgrund von $\lambda_i^+ = \lambda_i + \tilde{\rho}\mu_i$ ist (4.11) äquivalent zu

$$w_i(\mu) = 1 + \sum_{j=1}^{n} \frac{\tilde{u}_j^2}{\delta_{ji} - \mu} \quad , \tag{4.12}$$

mit $\delta_{ji} = \frac{\lambda_j - \lambda_i}{\tilde{\rho}}$.

(ii) Nach der Aufteilung von (4.12) in die Teilsummen

$$\psi(t) = \sum_{j=1}^{i} \frac{\tilde{u}_j^2}{\delta_{ji} - t} \text{ und } \phi(t) = \sum_{j=i+1}^{n} \frac{\tilde{u}_j^2}{\delta_{ji} - t} \tag{4.13}$$

ist $0 < \mu_i < \min(1 - \sum_{j=1}^{n-1} \mu_j, \delta_{ji})$ so zu bestimmen, dass $-\psi(\mu_i) = \phi(\mu_i) + 1$ gilt. Für die folgende iterative Berechnung von μ_i wird der Startwert t_1 so gewählt, dass $0 < t_1 < \mu_i$ ist.

(iii) Die Lösung von (ii) soll mit Hilfe der (4.13) interpolierenden, rationalen Funktionen

$$\psi_1 = \frac{p}{q - t_1}, \quad \phi_1 = r + \frac{s}{\delta_{j,i+1} - t_1}$$

geschehen. Dabei sind

$$\begin{aligned} p &= \frac{\psi_1^2}{\psi_1'}, \\ q &= t_1 + \frac{\psi_1}{\psi_1'}, \\ r &= \phi_1 - \phi_1'(\delta_{j,i+1} - t_1), \\ s &= (\delta_{j,i+1} - t_1)^2 \phi_1' \end{aligned}$$

und $\psi_1 = \psi(t_1)$ sowie $\phi_1 = \phi(t_1)$.

(iv) Die neue Approximation t_2 an μ_i ergibt sich aus

$$\frac{-p}{q-t_2} = 1 + r + \frac{s}{\delta_{j,i+1} - t_2}.$$

Die Lösung dieser quadratischen Gleichung führt auf folgende explizite Darstellung von t_2:

$$t_2 = t_1 + 2\frac{b}{a + \sqrt{a^2 - 4b}}.$$

Hierbei sind

$$\begin{aligned} a &= \frac{(\delta_{j,i+1} - t_1)(1+\phi_1) + \frac{\psi_1^2}{\psi_1'}}{c} + \frac{\psi_1}{\psi_1'}, \\ b &= \frac{(\delta_{j,i+1} - t_1) w \psi_1}{\psi_1' c}, \\ c &= 1 + \phi_1 - (\delta_{j,i+1} - t_1)\psi_1' \text{ und} \\ w &= 1 + \phi_1 + \psi_1 \quad . \end{aligned}$$

(v) Die Schritte (iii)-(iv) werden solange wiederholt, bis

$$|t_{i+1} - t_i| < \epsilon |t_i| \tag{4.14}$$

gilt, wobei mit ϵ die relative Maschinengenauigkeit bezeichnet ist. Alternative Abbruchkriterien werden in [14] diskutiert.

(vi) Der aufdatierte Eigenwert λ_i^+ kann letztendlich durch $\lambda_i^+ = \lambda_i + \tilde{\rho}\mu_i$ berechnet werden.

(vii) Mit (i)-(vi) lassen sich die Eigenwerte $\lambda_1^+, \cdots, \lambda_{n-1}^+$ berechnen. Der Fall $i = n$ wird wie folgt separat behandelt. Aus $i = n$ folgt, dass $\phi = 0$ ist. Damit reduziert sich (ii) zu $-\psi(t) = 1$. μ_n kann demnach durch die Iterationsvorschrift

$$t_{k+1} = t_k + \left(\frac{1+\psi_k}{\psi_k'}\right)\psi_k$$

berechnet werden.

In [14] wird bewiesen, dass diese Methode eine quadratische Konvergenzordnung besitzt.

Bemerkung 4.2.2. *In der hier beschriebenen Methode wird angenommen, dass $\tilde{\rho} > 0$ gilt. Dies stellt keine Beschränkung der Allgemeinheit dar, da im Fall von $\tilde{\rho} < 0$, λ_i durch $-\lambda_{n-i+1}$ und $\tilde{\rho}$ durch $-\tilde{\rho}$ ersetzt werden können.*

4.2.3 Berechnung der aufdatierten Eigenvektoren

Sind die Eigenwerte $\lambda_1^+, \ldots, \lambda_n^+$ von B_+ bekannt, ist zur Berechnung der jeweilig zugehörigen Eigenvektoren $Q_+ = \left[q_1^+, \ldots, q_n^+\right]$, mit $q_i^+ \in \mathbb{R}^n$ je ein Gleichungssystem der Form

$$B_+ q_i^+ - \lambda_i^+ q_i^+ = 0, \text{ für } 1 \leq i \leq n$$

zu lösen.

Mit $\hat{u} = Q^T u$ und $D_i = D - \lambda_i^+ I$ sind die aufdatierten Eigenvektoren q_i^+ somit durch

$$q_i^+ = \frac{Q D_i^{-1} \hat{u}}{\|D_i^{-1} \hat{u}\|_2}, \text{ für } 1 \leq i \leq n \tag{4.15}$$

explizit berechenbar. Die Lösung eines jeden solchen Gleichungssystems erfordert einen Rechenaufwand der Größenordnung $\mathcal{O}(n^2)$. Damit sind für die gesamte Berechnung der Matrix Q_+, durch dieses in [14] vorgeschlagene Vorgehen, $\mathcal{O}(n^3)$ Operationen nötig. Da die direkte Berechnung der Singulärwertzerlegung der aufdatierten Matrix eine ebensolche Komplexität aufweist, ist es, den Rechenaufwand betrachtet, nicht von Vorteil die Eigenvektoren durch Lösen der Gleichungssysteme (4.15) zu berechnen. Weitaus effizientere Methoden werden im folgenden Kapitel 4.3 behandelt.

4.3 Multiplikation mit Cauchy-Matrizen

In den folgenden Abschnitten werden verschiedene Möglichkeiten der effizienten Multiplikation von Matrix-Vektor Produkten der Form Cy, wobei $y \in \mathbb{R}^n$ und $C \in \mathbb{R}^{n \times n}$ eine Cauchy-Matrix gemäß (2.23) ist, behandelt. Diese Problemstellung wird auch als Trummer-Problem [27] bezeichnet. Durch die spezielle Struktur der Matrix C kann diese Multiplikation bestenfalls mit einer Komplexität von $\mathcal{O}(n \log^2 n)$, im Gegensatz zu $\mathcal{O}(n^2)$ im Fall allgemeiner Matrizen, durchgeführt werden.

4.3.1 Der erweiterte GGS-Algorithmus

Eine erste Möglichkeit zur effizienten Multiplikation von Cauchy-Matrizen mit Vektoren stellt die in [23] präsentierte Erweiterung des GGS-Algorithmus aus [24] dar.

Dieses Verfahren basiert darauf, dass die Multiplikation $f = C_{(s,t)} y \in \mathbb{R}^n$ äquivalent zur Auswertung der Funktion

$$f(x) = \sum_{j=1}^{n} \frac{y_j}{x - s_j}$$

an den Punkten t_i mit $i = 1, \ldots, n$ ist. $C_{(s,t)}$ bezeichnet analog zu Kapitel 2.6 die aus den Vektoren s und t erzeugte Cauchy-Matrix.

Nach [22] kann $f(x)$ als Quotient zweier Polynome $h(x)$ und $g(x)$ darstellt werden. Dabei ist $g(x)$ vom Grad n gegeben durch

$$g(x) = \prod_{j=1}^{n}(x - s_j).$$

Das Polynom $h(x)$, welches maximal vom Grad $n-1$ ist, wird aus dem Quotienten

$$f(x) = \frac{h(x)}{g(x)} = \sum_{j=1}^{n} \frac{y_j}{x - s_j} \tag{4.16}$$

abgeleitet. Setzt man dazu $x = s_i$ für $i = 1, \ldots, n$ in (4.16) ein, ergibt sich

$$h(s_i) = y_i g'(s_i), \quad i = 1, \ldots, n.$$

Demzufolge ist $h(x)$ genau das Interpolationspolynom der Punkte $(s_i, y_i g'(s_i))$.

Folgender Algorithmus ermöglicht die effiziente Berechnung von $f(t_i) = \frac{h(t_i)}{g(t_i)}$:

Erweiterter GGS-Algorithmus

(1) Berechnung der Koeffizienten von $g(x)$ mit Hilfe der FFT Polynommultiplikation. $\mathcal{O}(n \log^2 n)$ [45]

(2) Berechnung der Koeffizienten von $g'(x)$. $\mathcal{O}(n)$

(3) Berechnung von $g(t_i)$ und $g'(s_j)$ mit $i, j = 1, \ldots, n$. $\mathcal{O}(n \log^2 n)$ [1]

(4) Berechnung von $h_j = y_j g'(s_j)$, $j = 1, \ldots, n$. $\mathcal{O}(n)$

(5) Berechnung des Interpolationspolynoms $h(x)$ für die Punkte (s_j, h_j), $j = 1, \ldots, n$. $\mathcal{O}(n \log^2 n)$ [1]

(6) Berechnung von $h(t_i)$. $\mathcal{O}(n)$

(7) Berechnung von $Cy := f(t_i) = \frac{h(t_i)}{g(t_i)}$. $\mathcal{O}(n)$

Der hier beschriebene Algorithmus zeigt, dass es möglich ist, ein Produkt aus Cauchy-Matrix und Vektor mit $\mathcal{O}(n \log^2 n)$ Operationen zu berechnen. Dessen praktische Anwendbarkeit ist im Allgemeinen allerdings auf Probleme kleiner Dimension n beschränkt. Dies ist durch die in Schritt (5) auftretende Berechnung des Interpolationspolynoms $h(x)$ bedingt. Bei großer Dimension n und ebenso im Fall von ungünstig verteilten Stützstellen s_i sowie Funktionswerten $y_i g'(s_i)$ führt diese Interpolation zu numerischer Instabilität.

4.3.2 Transformation von Cauchy-Matrizen

Eine Möglichkeit, die bei ungünstigen Knotenverteilungen auftretenden Stabilitätsprobleme des erweiterten GGS-Algorithmus zu umgehen, ist die Transformation der ursprünglichen Cauchy-Matrix. In [61] ist gezeigt, dass sich eine Cauchy-Matrix mit den Basisvektoren s und t in ein Produkt von Diagonalmatrizen sowie zweier Cauchy-Matrizen mit den jeweiligen Basisvektoren s und b sowie t und b zerlegen lässt. Hierbei kann der Vektor $b \in \mathbb{R}^n$ beliebig gewählt werden.

Definition 4.3.1. *Durch die Vektoren $a, b \in \mathbb{R}^n$ seien folgende Polynome definiert:*

$$\begin{aligned} p_b(x) &= \prod_{j=1}^{n}(x - b_j) \quad , \\ p'_b(x) &= \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} (x - b_j) \quad . \end{aligned}$$

Weiterhin seien die durch $D_{(a,b)}, D'_{(b)} \in \mathbb{R}^{n\times n}$ bezeichneten Diagonalmatrizen wie folgt definiert:

$$\begin{aligned} D_{(a,b)} &= \operatorname{diag}(p_b(a_i))_{i=1}^n = \operatorname{diag}(\prod_{j=1}^{n}(a_i - b_j))_{i=1}^n \quad \textit{sowie} \\ D'_{(b)} &= \operatorname{diag}(p'_b(b_j))_{j=1}^n = \operatorname{diag}(\prod_{j=1, j\neq i}^{n} (a_i - b_j))_{i=1}^n \quad . \end{aligned}$$

Theorem 4.3.1. *Sei $C_{(t,s)} \in \mathbb{R}^{n\times n}$ die durch die Vektoren $t \in \mathbb{R}^n$ und $s \in \mathbb{R}^n$ definierte Cauchy-Matrix, sowie $b \in \mathbb{R}^n$ ein beliebig gewählter Vektor. Dann kann die Cauchy-Matrix $C_{(t,s)}$ durch die Faktorisierung*

$$C_{(t,s)} = -C_{(t,b)} D_{(b,t)} D'^{-1}_{(b)} C_{(b,s)} D_{(s,b)} D^{-1}_{(s,t)} \tag{4.17}$$

dargestellt werden.

Beweis. Siehe [61]. □

Darstellung (4.17) ermöglicht es, durch eine geeignete Wahl des Vektors b die Stabilitätseigenschaften des erweiterten GGS-Algorithmus in Bezug auf die zu multiplizierenden Cauchy-Matrizen zu verbessern.

Die Multiplikation von $C_{(t,s)}$ mit einem Vektor $y \in \mathbb{R}^n$ ist auch unter Verwendung der rechten Seite von (4.17) in einer Komplexität von $\mathcal{O}(n \log^2 n)$ möglich. Allerdings müssen in diesem Fall drei Produkte der Form Diagonalmatrix mal Vektor sowie zwei Produkte

der Form Cauchy-Matrix mal Vektor gebildet werden. Aus diesem Grund sowie durch die Berechnung der Diagonalmatrizen erhöht sich die Konstante in $\mathcal{O}(n \log^2 n)$ um ein Vielfaches. Da auch die geeignete Bestimmung des Vektors b eine nicht-triviale Aufgabe darstellt, hat sich die hier vorgestellte Transformation in den wenigsten Beispielen als praktikabel erwiesen, um auftretende Instabilitäten bei der Benutzung des erweiterten GGS-Algorithmus zu beseitigen.

4.3.3 Effiziente Multiplikation durch Matrixapproximation

Eine weitere Möglichkeit zur effizienten Multiplikation von Cauchy-Matrizen $C_{(t,s)}$ mit Vektoren ist die Verwendung hierarchischer Matrixapproximationstechniken. Diese Idee beruht darauf große Matrizen zunächst mit Hilfe einer Niedrigrangdarstellung zu approximieren. Mit Hilfe dieser Darstellung können Matrixoperationen mit weniger Operationen durchgeführt werden, als wenn dies mit der ursprünglichen Matrix geschehen würde. Eine umfassende Darstellung von $\mathcal{H}$-Matrizen ist unter anderem in [31, 41, 43] gegeben.

Im Folgenden wird zunächst gezeigt, dass sich Cauchy-Matrizen sehr gut durch Niedrigrangmatrizen approximieren lassen.

Konstruktion der Niedrigrangapproximation

Dem Vorgehen aus [32] folgend, wird zunächst die separable Darstellung

$$f(x,y) = \frac{1}{x-y} \approx \sum_{l=1}^{k} g_l(x) h_l(y)$$

genutzt, um die Elemente von $C_{(t,s)} = (C_{i,j})_{1 \le i,j \le n}$ zu approximieren. Es sei dazu

$$C_{i,j} \approx \sum_{l=1}^{k} g_l(t_j) h_l(s_i). \tag{4.18}$$

Für die folgenden Betrachtungen sollen die Elemente der Vektoren t und s als die abgeschlossenen, n-elementigen Teilmengen

$$T_{(1,n)} = \{t_1, \ldots, t_n\} \text{ sowie } S_{(1,n)} = \{s_1, \ldots, s_n\} \tag{4.19}$$

der reellen Zahlen dargestellt werden. Hierbei wird angenommen, dass die einzelnen Elemente der Vektoren bereits der Größe nach geordnet sind, d.h.

$$\begin{aligned} t_1 &\le t_2 \le \cdots \le t_n \\ s_1 &\le s_2 \le \cdots \le s_n. \end{aligned}$$

Mit $t_1 = \lambda_n, \ldots, t_n = \lambda_1$ und $s_1 = \lambda_n^+, \ldots, s_n = \lambda_1^+$ entspricht dies genau der Sortierung der Eigenwerte des symmetrischen Eigenwertproblems in Kapitel 4.2. Aufgrund von (4.6) sowie der Deflation in Kapitel 4.2.1 folgt weiterhin, dass die Elemente der Mengen $S_{(1,n)}$ und $T_{(1,n)}$ in der in Kapitel 4.5.4 betrachteten Anwendung disjunkt zueinander sind.

Definition 4.3.2. *Durch*

$$\begin{aligned} \operatorname{diam}(T_{(1,n)}) &= \max(T_{(1,n)}) - \min(T_{(1,n)}), \\ \operatorname{diam}(S_{(1,n)}) &= \max(S_{(1,n)}) - \min(S_{(1,n)}) \end{aligned} \tag{4.20}$$

mit $\max(T_{(1,n)}) = \max\limits_{1\le i\le n}(t_i)$, $\max(S_{(1,n)}) = \max\limits_{1\le i\le n}(s_i)$, $\min(T_{(1,n)}) = \min\limits_{1\le i\le n}(t_i)$ *und* $\min(S_{(1,n)}) = \min\limits_{1\le i\le n}(s_i)$ *wird der Durchmesser von* $T_{(1,n)}$ *bzw.* $S_{(1,n)}$ *bezeichnet.*

Weiterhin sei der Abstand zwischen den Mengen $T_{(1,n)}$ *und* $S_{(1,n)}$ *durch*

$$\operatorname{dist}(S_{(1,n)}, T_{(1,n)}) = \begin{cases} \min(T_{(1,n)}) - \max(S_{(1,n)}), & \textit{falls } \min(T_{(1,n)}) > \max(S_{(1,n)}) \\ \min(S_{(1,n)}) - \max(T_{(1,n)}), & \textit{falls } \min(S_{(1,n)}) > \max(T_{(1,n)}) \\ 0, & \textit{sonst} \end{cases} \tag{4.21}$$

definiert.

Die Approximation (4.18) ist mit niedrigem Rang k möglich, wenn die Bedingung

$$\min\left(\operatorname{diam}(S_{(1,n)}), \operatorname{diam}(T_{(1,n)})\right) \le \operatorname{dist}(S_{(1,n)}, T_{(1,n)}) \tag{4.22}$$

erfüllt ist.

Ist Bedingung (4.22) erfüllt und sei weiterhin

$$\hat{t} = \frac{\min(T_{(1,n)}) + \max(T_{(1,n)})}{2}, \quad \hat{s} = \frac{\min(S_{(1,n)}) + \max(S_{(1,n)})}{2},$$

dann kann jedes Element $C_{i,j}$ der Cauchy-Matrix $C_{(t,s)}$ durch

$$\hat{c}_{i,j} = \begin{cases} \sum\limits_{l=0}^{k} (\hat{t} - t_j)^{-l-1} (\hat{t} - s_i)^l, & \text{falls } \operatorname{diam}(T_{(1,n)}) \le \operatorname{diam}(S_{(1,n)}) \\ \sum\limits_{l=0}^{k} (s_i - \hat{s})^{-l-1} (t_j - \hat{s})^l, & \text{sonst} \end{cases} \tag{4.23}$$

approximiert werden. Mit $\hat{C}_k = (\hat{c}_{i,j})_{1\leq i,j\leq n}$ sei die Approximation von $C_{(t,s)}$ vom Rang k bezeichnet. Aus (4.23) lassen sich weiterhin mit

$$\begin{aligned} K_{i,l} &= \begin{cases} (\hat{t}-s_i)^l, & \text{falls } \operatorname{diam}(T_{(1,n)}) \leq \operatorname{diam}(S_{(1,n)}) \\ (t_j-\hat{s})^l, & \text{sonst} \end{cases} \\ L_{j,l} &= \begin{cases} (\hat{t}-t_j)^{-l-1}, & \text{falls } \operatorname{diam}(T_{(1,n)}) \leq \operatorname{diam}(S_{(1,n)}) \\ (s_i-\hat{s})^{-l-1}, & \text{sonst} \end{cases} \end{aligned} \tag{4.24}$$

die Niedrigrangfaktoren

$$K = \underbrace{\begin{bmatrix} K_{1,1} & \cdots & K_{1,k} \\ \vdots & & \vdots \\ K_{n,1} & \cdots & K_{n,k} \end{bmatrix}}_{\in\mathbb{R}^{n\times k}} \quad \text{und} \quad L = \underbrace{\begin{bmatrix} L_{1,1} & \cdots & L_{1,k} \\ \vdots & & \vdots \\ L_{n,1} & \cdots & L_{n,k} \end{bmatrix}}_{\in\mathbb{R}^{n\times k}}$$

der Zerlegung $\hat{C}_k = KL^T$ ableiten.

Lemma 4.3.2. *Soll für die elementweise Approximation der Cauchy-Matrix*

$$|\hat{c}_{i,j} - C_{i,j}| \leq \epsilon |C_{i,j}|$$

mit $0 < \epsilon < 1$ *gelten, so muss der Rang* k *der Niedrigrangfaktoren* K *und* L *mindestens*

$$k = \left\lceil \log_3 \left(\frac{1}{\epsilon} \right) \right\rceil + 1 \tag{4.25}$$

sein.

Beweis. Siehe [32]. □

Bemerkung 4.3.1. *Um eine Approximation in Größenordung der Maschinengenauigkeit* $\epsilon \approx 10^{-16}$ *zu erhalten, ist demnach ein relativ hoher Rang von* $k = 35$ *nötig.*

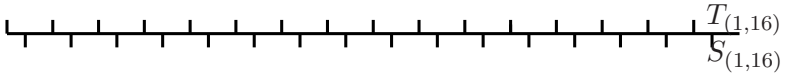
Zerlegung der Cauchy-Matrix in approximierbare Blöcke

In dieser Arbeit ist die Approximierbarkeit der orthogonalen Matrizen, welche durch die symmetrische Niedrigrangaufdatierung des Eigenwertproblems in Kapitel 4.2 entstehen, von besonderem Interesse. Für diese ist die Bedingung (4.22) nicht erfüllt, da zusätzlich zu (4.19) auch

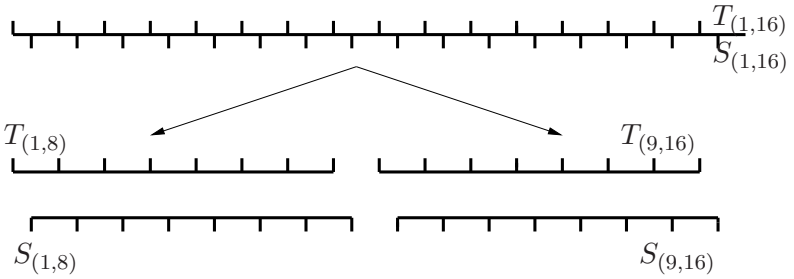
$$t_1 \leq s_1 \cdots \leq t_n \leq s_n \tag{4.26}$$

gilt. Um die zuvor eingeführte Approximation nutzen zu können, ist zunächst eine Unterteilung von $T_{(1,n)}$ sowie $S_{(1,n)}$ in Teilmengen, welche die Bedingung (4.22) erfüllen, nötig. Verschiedene Konstruktionen hierzu sind zum Beispiel in [43] beschrieben.

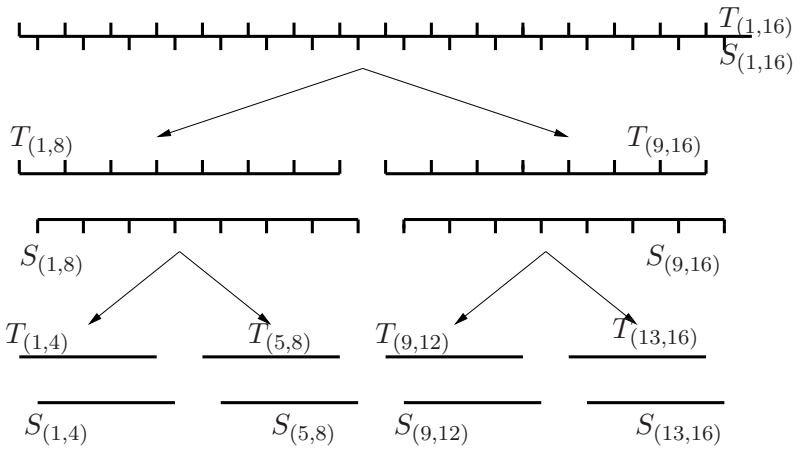
Ein mögliches Vorgehen soll im Folgenden anhand eines Beispiels beschrieben werden. Der Einfachheit halber wird davon ausgegangen, dass die Elemente der Mengen $T_{(1,16)} = \{1, \ldots, 16\}$ sowie $S_{(1,16)} = \{1.4, 2.4, \ldots, 16.4\}$ wie folgt auf der reellen Achse verteilt sind:



Da $\operatorname{diam}(S_{(1,16)}) = \operatorname{diam}(T_{(1,16)}) = 15$ und $\operatorname{dist}(T_{(1,16)}, S_{(1,16)}) = 0$ sind, ist Bedingung (4.22) verletzt. Um eine Niedrigrangapproximation gemäß des vorhergehenden Kapitels durchführen zu können, müssen die Mengen zunächst so unterteilt werden, dass die neuen Teilmengen (4.22) erfüllen. In diesem Beispiel soll dazu eine sich wiederholende Intervallhalbierung vorgenommen werden. Nach dem ersten Schritt erhält man demzufolge:



Da noch immer für keine der Kombinationen $T_{(1,8)}$, $T_{(9,16)}$ mit $S_{(1,8)}$, $S_{(9,16)}$ die Bedingung (4.22) erfüllt ist, muss eine weitere Unterteilung vorgenommen werden:



Nach der zweiten Bisektion erfüllen folgende Mengen die Bedingung (4.22):

$$\begin{aligned}
\min(\operatorname{diam}(S_{(5,8)}), \operatorname{diam}(T_{(13,16)}) &\leq \operatorname{dist}(S_{(5,8)}, T_{(13,16)}) \\
\min(\operatorname{diam}(S_{(9,12)}), \operatorname{diam}(T_{(1,4)}) &\leq \operatorname{dist}(S_{(9,12)}, T_{(1,4)}) \\
\min(\operatorname{diam}(S_{(1,4)}), \operatorname{diam}(T_{(9,16)}) &\leq \operatorname{dist}(S_{(1,4)}, T_{(9,16)}) \qquad (4.27) \\
\min(\operatorname{diam}(S_{(13,16)}), \operatorname{diam}(T_{(1,8)}) &\leq \operatorname{dist}(S_{(13,16)}, T_{(1,8)}). \qquad (4.28)
\end{aligned}$$

Hierbei wurden in (4.27) und (4.28) jeweils zwei Teilmengen von T zusammengefasst, um möglichst große Matrixblöcke zu erzeugen, welche approximiert werden können. Für die zugehörige Matrix $C_{(t,s)}$ folgt nach dieser zweiten Unterteilung, dass folgende schraffierte Blöcke durch Niedrigrang approximiert werden können:

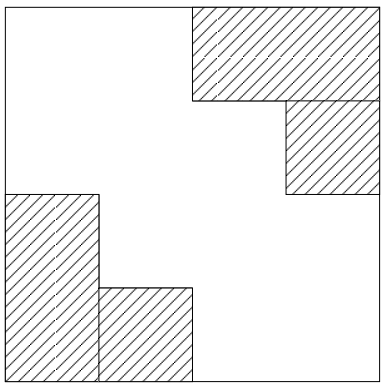
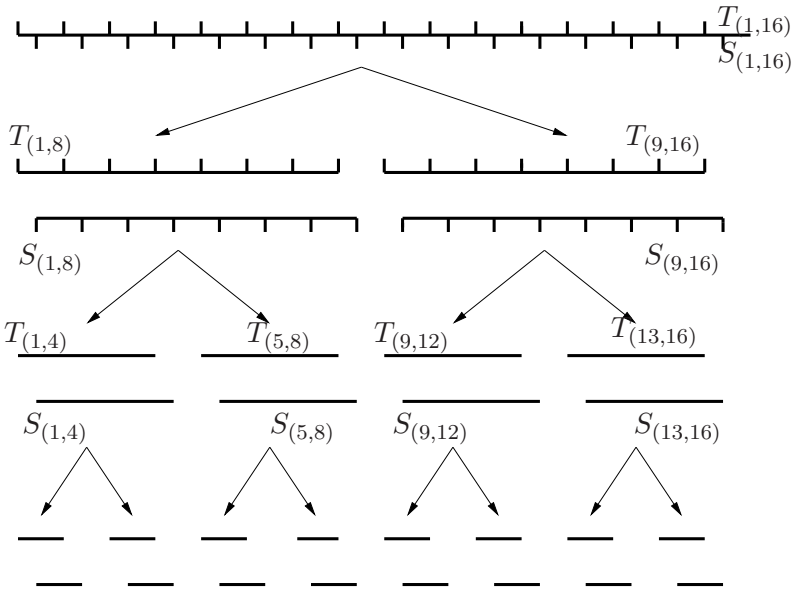


Abb. 4.3.1: Blockzerlegung der Cauchy-Matrix im 2. Schritt

Für den verbleibenden, unschraffierten Teil müssen weitere Intervallunterteilungen durchgeführt werden. Letztendlich ergibt sich in diesem Beispiel



mit der zugehörigen Matrix:

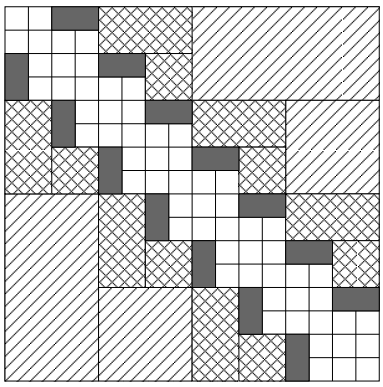


Abb. 4.3.2: Blockzerlegung der Cauchy-Matrix

Abbildung 4.3.2 zeigt die typische Blockstruktur, welche bei der Approximation der durch die Rang-1 Aufdatierung des symmetrischen Eigenwertproblems entstehenden Cauchy-Matrizen, vgl. Kapitel 4.2, auftritt.

Bemerkung 4.3.2. *Die angenommene, äquidistante Verteilung der Elemente von $T_{(1,16)}$ und $S_{(1,16)}$ dient lediglich zur besseren Veranschaulichung des beschriebenen Vorgehens. In Anwendungen tritt diese praktisch nicht auf. Demzufolge ist auch die einfache Bisektion der Mengen in den meisten Fällen nicht die geeignetste Variante, um möglichst große, approximierbare Blöcke zu erzeugen.*

Bemerkung 4.3.3. *Die Niedrigrangapproximation eines Blocks ist natürlich nur dann*

sinnvoll, wenn der für die gewünschte Genauigkeit ϵ erforderliche Rang k, vgl. (4.25), kleiner ist, als der Rang des Matrixblocks selbst.
Durch die Einführung eines Paramters $\eta > 0$ in (4.22):

$$\min\left(\operatorname{diam}(S_{(1,n)}), \operatorname{diam}(T_{(1,n)})\right) \leq \eta \cdot \operatorname{dist}(S_{(1,n)}, T_{(1,n)}) \tag{4.29}$$

ist es möglich, die Größe der zu approximierenden Blöcke zu steuern. Ein solches Vorgehen wird typischerweise bei $\mathcal{H}$-Matrizen angewendet, siehe [43].

Bemerkung 4.3.4. *Da die Niedrigrangdarstellung der Matrix $C_{(t,s)}$ nur zur effizienten Multiplikation mit Vektoren benötigt wird, stellen die rechteckigen Blöcke, vgl. Abbildung 4.3.2, kein Problem dar. Diese versucht man im Fall von quadratischen Matrizen typischerweise zu vermeiden, damit Matrix-Matrix Produkte der Art $C_{(t,s)} \cdot C_{(t,s)}$ in formatierter Form effizient berechenbar sind. Siehe hierzu auch [43].*

Multiplikation mit Vektoren

Die in den vorangehenden Abschnitten beschriebene Niedrigrangapproximation der Matrix $C_{(t,s)}$ ermöglicht eine effiziente Matrix-Vektor Multiplikation $\hat{y} = C_{(t,s)} \cdot y$.

Sei $C \in \mathbb{R}^{n \times m}$ eine vollbesetzte Matrix und $y \in \mathbb{R}^m$. Dann sind zur Berechnung von $\hat{y} = Cy$ genau nm Multiplikationen sowie $n(m-1)$ Additionen nötig. Wird die Matrix C durch das Produkt der Niedrigrangfaktoren $K \in \mathbb{R}^{n \times k}$ und $L \in \mathbb{R}^{k \times m}$ vom Rang k approximiert, so entspricht das Matrix-Vektor Produkt Cy dem Ergebnis der Multiplikationen $\tilde{y} = L^T y$ gefolgt von $\hat{y} = K\tilde{y}$. Hierbei ergibt sich ein Rechenaufwand von $k(n+m)$ Multiplikationen und $k(m+n-1) - n$ Additionen. Daraus folgt, dass die Verwendung der Niedrigrangdarstellung für Ränge

$$k < \frac{nm}{n+m}$$

bei der Berechnung des Produkts Cy effizienter ist. Es folgt, dass, je kleiner der Rang k ist, desto mehr Blöcke der Cauchy-Matrix $C_{(t,s)}$ sinnvollerweise durch Niedrigrang approximiert werden können. Hierbei darf allerdings der Aufwand, welcher zur Unterteilung der Mengen $T_{(1,n)}$ und $S_{(1,n)}$ sowie zur Berechnung der Niedrigrangfaktoren K und L nötig ist, nicht außer Acht gelassen werden.

Die eigentliche Multiplikation wird nach folgendem Schema durchgeführt. Sei

$$C = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

eine Matrix, deren Blöcke $C_2 = K_2 L_2^T \in \mathbb{R}^{n\times n}$ und $C_3 = K_3 L_3^T \in \mathbb{R}^{n\times n}$ in Niedrigrangdarstellung vom Rang $k \ll n$ vorliegen. Die beiden Matrizen $C_1 \in \mathbb{R}^{n\times n}$ und $C_4 \in \mathbb{R}^{n\times n}$ seien vollbesetzt dargestellt. Dann wird die Multiplikation

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{4.30}$$

in folgender Weise durchgeführt:

$$\begin{aligned} \hat{y}_1 &= \overbrace{C_1 y_1}^{\mathcal{O}(n^2)} + \overbrace{K_2(L_2^T y_2)}^{\mathcal{O}(kn)} \\ \hat{y}_2 &= \underbrace{K_3(L_3^T y_1)}_{\mathcal{O}(kn)} + \underbrace{C_4 y_2}_{\mathcal{O}(n^2)}. \end{aligned} \tag{4.31}$$

Auf weitere, mehrfache Unterteilungen der Matrix C kann (4.31) rekursiv angewendet werden.

Beispiel 4.3.1. *Sei $C \in \mathbb{R}^{1024\times 1024}$ eine Cauchy-Matrix. Die Verteilung der Elemente in den zugehörigen Mengen $T_{(1,1024)}$ sowie $S_{(1,1024)}$ sei, wie im Schema zuvor, äquidistant. Die Erzeugung approximierbarer Blöcke ist durch Intervallhalbierung geschehen. Die kleinsten zur Niedrigrangapproximation zugelassenen Blöcke seien von der Dimension* (64×64)*. Der verwendete Rang sei $k = 30$, was einer Genauigkeit von $\epsilon = 10^{-14}$ entspricht. In diesem Fall ist die Anzahl der benötigten Multiplikationen zur Berechnung von $\hat{y} = Cy \in \mathbb{R}^{1024}$ gleich* 512464*. Im Gegensatz dazu wären bei der standardmäßigen Berechnung des Produkts* 1048576 *Multiplikationen nötig. Für die Anzahl der Additionen gilt Ähnliches. Durch die Niedrigrangapproximation kann somit der Rechenaufwand in diesem Fall um ca. 50% reduziert werden.*

4.3.4 Approximation durch Exponentialsummen

Der Nachteil der im vorherigen Kapitel 4.3.3 beschriebenen Konstruktion ist der relativ hohe Rang k, welcher durch sehr kleine Approximationsfehler ϵ festgelegt wird. Eine zu (4.18) alternative, separable Darstellung von $C_{(t,s)}$ ist mit Hilfe von Exponentialfunktionen möglich. Im Folgenden werden einige Ergebnisse der Arbeiten [11, 42] vorgestellt und auf den hier vorliegenden Fall angepasst.

Approximation von x^{-1}

Theorem 4.3.3. *In einem festen Intervall $[a, b]$ mit $0 < a < b$ kann die Funktion $f(x) = \frac{1}{x}$ durch eine eindeutig definierte Exponentialsumme der Form*

$$s_k(x) = \sum_{l=1}^{k} \omega_i e^{-a_i x}, \quad \omega_i, a_i \in \mathbb{R}_+ \tag{4.32}$$

approximiert werden, so dass die Chebychev Norm

$$\|f - s_k\|_{\infty,[a,b]} = \max_{a \le x \le b} (|f(x) - s_k(x)|)$$

minimal ist.

Beweis. Siehe [10]. □

Lemma 4.3.4. *Sei s_k^* die Bestapproximation der Form (4.32) an $f(x)$ im Intervall $[a, b]$. Dann fällt der Fehler $\epsilon_k = \|f - s_k^*\|_{\infty,[a,b]}$ gemäß $\epsilon_k \le C_1 e^{-C_2 k}$ für $k \to \infty$ exponentiell ab.*

Beweis. Siehe [11]. □

Die in der folgenden Tabelle auf zwei Nachkommastellen genau angegebenen Werte wurden in [11] für die Konstanten C_1 sowie C_2 für verschiedene Ränge k numerisch bestimmt:

k	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
C_1	4.89	10.26	12.30	13.72	14.83
C_2	4.01	4.29	4.33	4.35	4.36

Tabelle 4.3.1: Konstanten C_1 und C_2 in Abhängigkeit vom Rang k

Die Konstanten C_1 und C_2 hängen sowohl vom Rang k der Approximation als auch von der Intervalllänge $b - a$ ab. Aus diesem Grund ist in der obigen Tabelle nur ein kleiner Ausschnitt dieser Werte gegeben. Eine umfangreiche Sammlung dazu bietet [42].

Bemerkung 4.3.5. *x^{-1} als auch $e^{-a_i x}$ können problemlos skaliert werden. Zur Vereinfachung der Notation wird deswegen im Folgenden an Stelle des Intervalls $[a, b]$ das neue Intervall $[1, R]$ verwendet, wobei $R = \frac{b}{a}$ ist.*

Die Approximation (4.32) lässt sich wie folgt auf die multivariante Funktion $f(x_1, \ldots, x_n)$ anwenden:

$$\frac{1}{x_1 + \cdots + x_n} \approx \sum_{i=1}^{k} \left(\omega_i \prod_{j=1}^{n} e^{-a_i x_j} \right). \tag{4.33}$$

Diese Eigenschaft wird es im Weiteren ermöglichen, ähnlich zu (4.18), bestimmte Blöcke der Cauchy-Matrix $C_{(t,s)}$ mit Hilfe der Darstellung (4.33) zu approximieren.

Hierzu wird zunächst davon ausgegangen, dass bereits eine Partitionierung der die Cauchy-Matrix definierenden Mengen $T_{(1,n)}$ sowie $S_{(1,n)}$ gemäß Kapitel 4.3.3 vorgenommen wurde. Seien im Folgenden durch $S = \{s_i\}_{1 \le i \le m}$ und $T = \{t_j\}_{1 \le j \le n}$ zwei Mengen bezeichnet, welche Bedingung (4.22) erfüllen. Dann gelten die Ungleichungen

$$\begin{aligned} \min(S) - \max(T) &\le s_i - t_j \le \max(S) - \min(T) \\ \operatorname{dist}(S,T) &\le s_i - t_j \le \operatorname{diam}(S) + \operatorname{diam}(T) + \operatorname{dist}(S,T) \\ 1 &\le s_i - t_j \le 1 + \frac{\operatorname{diam}(S) + \operatorname{diam}(T)}{\operatorname{dist}(S,T)} \end{aligned} \tag{4.34}$$

für alle $i = 1, \ldots, m$ und $j = 1, \ldots, n$.

Aus (4.34) ist zu erkennen, dass die Differenz der Elemente von S und T in einem relativ kleinen Intervall liegen. Daraus folgt, dass die zugehörigen Elemente $\frac{1}{s_i - t_j}$ in diesem Block der Cauchy-Matrix zur Approximation durch die Exponentialsumme

$$\frac{1}{s_i - t_j} = \sum_{l=1}^{k} \omega_l e^{-a_l s_i} e^{a_l t_j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{4.35}$$

geeignet sind. Aufgrund des positiven Exponenten $a_i t_j$ in der rechten Summe der Gleichung (4.35) treten bei deren Auswertung allerdings numerische Probleme auf. Diese können vermieden werden, wenn anstelle von $\frac{1}{s_i - t_j}$ die Brüche

$$\frac{\operatorname{dist}(S,T)}{\underbrace{(s_i - w)}_{=\tilde{s}_i > 0} + \underbrace{(w - t_j)}_{=\tilde{t}_j > 0}}, \quad \text{mit } w = \frac{\min(S) - \max(T)}{2}$$

approximiert werden. Da $\tilde{s}_i > 0$ sowie $\tilde{t}_j > 0$ ist, folgt, dass beide Exponenten in (4.35) negativ sind. Die Skalierung mit dem Faktor $\operatorname{dist}(S,T)$ reduziert die Länge des zu approximierenden Intervalls $[1, R]$ und ermöglicht somit kleinere Ränge k.

Mit Hilfe dieser Transformationen sowie der Formel (4.35) ist folgende Niedrigrangapproximation von Blöcken der Matrix $C_{(t,s)}$, für welche Bedingung (4.22) erfüllt ist,

möglich:

$$
\begin{aligned}
C &= \left(\frac{1}{s_i - t_j}\right)_{1\le i\le m, 1\le j\le n} \\
&= \frac{1}{d} \underbrace{\begin{bmatrix} \omega_1 e^{-\frac{1}{d}a_1\tilde{s}_1} & \dots & \omega_k e^{-\frac{1}{d}a_k\tilde{s}_1} \\ \vdots & & \vdots \\ \omega_1 e^{-\frac{1}{d}a_1\tilde{s}_m} & \dots & \omega_k e^{-\frac{1}{d}a_k\tilde{s}_m} \end{bmatrix}}_{=K\in\mathbb{R}^{m\times k}} \underbrace{\begin{bmatrix} e^{-\frac{1}{d}a_1\tilde{t}_1} & \dots & e^{-\frac{1}{d}a_1\tilde{t}_n} \\ \vdots & & \vdots \\ e^{-\frac{1}{d}a_k\tilde{t}_1} & \dots & e^{-\frac{1}{d}a_k\tilde{t}_n} \end{bmatrix}}_{=L^T\in\mathbb{R}^{k\times n}}.
\end{aligned}
$$

Hierbei ist $d = \operatorname{dist}(S, T)$. Mit Hilfe dieser Niedrigrangfaktoren K und L kann das Matrix-Vektor Produkt $\hat{y} = Cy \approx KL^T y$, wie bereits in Kapitel 4.3.3 gezeigt, effizient berechnet werden.

Aus der beschriebenen Konstruktion der Approximation durch Exponentialsummen sowie der Bedingung (4.22) folgt, dass das zu approximierende Intervall, vgl. Bemerkung 4.3.5, durch $[1, R] = [1, 3]$ beschränkt ist. Nach [42] folgt daraus, dass für einen Approximationsfehler der Größenordnung $\epsilon_k = 10^{-16}$ der Rang $k = 11$ nötig ist. Im Vergleich dazu muss für diese Genauigkeit im Fall der Approximation mittels (4.24) ein Rang von 35 verwendet werden. Die Vorteile des deutlich reduzierten Rangs liegen zum einen darin, dass auch die Approximation kleiner Blöcke, welche Bedingnung (4.22) erfüllen, die Berechnung des Matrix-Vektor Produkts beschleunigen. Zum anderen werden bei der Multiplikation mit den approximierten Blöcken deutlich weniger Operationen benötigt.

Beispiel 4.3.2. *Es gelten die Voraussetzungen aus Beispiel 4.3.1. Da bei der Approximation durch Exponentialsummen nur der Rang $k = 11$ nötig ist, um Maschinengenauigkeit zu erreichen, sollen die kleinsten zur Approximation zugelassenen Blöcke die Dimension (32×32) haben. Unter diesen Vorraussetzungen erfordert die Matrix-Vektor Multiplikation $\hat{y} = Cy \in \mathbb{R}^{1024}$ nur* 263172 *Multiplikationen. Dies entspricht ungefähr 50% der Operationen, welche bei der Approximation durch (4.24) und 25% der Operationen, welche bei standardmäßiger Matrix-Vektor Multiplikation nötig wären. Für die Anzahl der Additionen gilt dies ebenfalls.*

Bemerkung 4.3.6. *Die Berechnung der Konstanten a_l und ω_l in (4.35) ist sehr aufwendig. Aus diesem Grund muss bei der praktischen Anwendung der hier vorgestellten Approximation auf bereits bestehende Daten [42] zugegriffen werden.*

4.4 Aufdatierung im unsymmetrischen Fall

Gegeben sei die Singulärwertzerlegung

$$A = U\Sigma V^T \tag{4.36}$$

von $A \in \mathbb{R}^{m\times n}$. Dabei sind $U \in \mathbb{R}^{m\times m}$ sowie $V \in \mathbb{R}^{n\times n}$ orthogonale Matrizen deren Spalten den links- bzw. rechtsseitigen Singulärvektoren von A entsprechen. Sei weiterhin $\operatorname{rang}(A) = r$ sowie $\hat{\Sigma} = \operatorname{diag}(\sigma_i)_{1\leq i\leq r} \in \mathbb{R}^{r\times r}$ die Diagonalmatrix, deren Einträge die der Größe nach geordneten Singulärwerte, d.h. $\sigma_1 \geq \cdots \geq \sigma_r$, von A sind.

Damit ist die Matrix $\Sigma \in \mathbb{R}^{m\times n}$ durch

$$\Sigma = \begin{bmatrix} \hat{\Sigma} & 0_{r\times k} \\ 0_{l\times r} & 0_{l\times k} \end{bmatrix}, \tag{4.37}$$

mit $k = n - r$ und $l = m - r$ definiert.

Für die folgenden Betrachtungen soll der Einfachheit halber angenommen werden, dass A vollen Rang hat. Damit ist $r = m$ und (4.37) reduziert sich zu

$$\Sigma = \begin{bmatrix} \hat{\Sigma} & 0_{m\times k} \end{bmatrix}.$$

Weiterhin seien $u \in \mathbb{R}^m$ und $v \in \mathbb{R}^n$ zwei Vektoren, welche die Rang-1 Modifikation

$$A_+ = A + uv^T \tag{4.38}$$

der Matrix A darstellen. Ziel ist die effiziente Berechnung der Singulärwertzerlegung

$$A_+ = U_+\Sigma_+V_+^T \quad . \tag{4.39}$$

Hierbei sind die Matrizen $U_+ \in \mathbb{R}^{m\times m}$, $V_+ \in \mathbb{R}^{n\times n}$ sowie $\Sigma_+ \in \mathbb{R}^{m\times n}$ analog zu (4.36)–(4.37) definiert.

Die explizite Neuberechnung von (4.39), ausgehend von (4.38), erfordert einen Rechenaufwand der Größenordnung $\mathcal{O}(n^2m)$, vgl. Kapitel 2.3. Zur Reduktion dieser kubischen Komplexität sollen U, V und Σ bezüglich der Rang-1 Aufdatierung (4.38) direkt modifiziert werden. Der im Folgenden vorgestellte, auf [65] basierende Algorithmus ermöglicht diese Berechnung in einer Größenordnung von $\mathcal{O}(n^2 \log^2 n)$ Operationen.

Im Gegensatz zum hier betrachteten allgemeinen Aufdatierungsproblem existieren zahlreiche effiziente Algorithmen für verschiedene Spezialfälle. Die Aufdatierung der symmetrischen Singulärwertzerlegung wird zum Beispiel in [14, 38] behandelt. Effiziente und stabile Algorithmen zur Aufdatierung für den Fall, dass die ursprüngliche Matrix A um eine Zeile erweitert bzw. das eine Zeile entfernt wird, sind unter anderem in [13, 37, 39] beschrieben.

4.5 Aufdatierungsverfahren

Der in dieser Arbeit beschriebene Algorithmus zur Rang-1 Aufdatierung der Singulärwertzerlegung besteht aus folgenden vier Schritten:

(1) Übergang vom rechteckigen Problem zu zwei symmetrischen Darstellungen.

(2) Berechnung der aufdatierten Singulärwerte.

(3) Berechnung der aufdatierten Singulärvektoren.

(4) Korrekturschritt zur Verbesserung der Genauigkeit der erhaltenen Lösung.

Diese vier Teilprobleme werden in den folgenden Kapiteln genauer dargestellt.

Bemerkung zur Notation
Aufdatierte Matrizen werden im Folgenden mit dem Index '+' gekennzeichnet. Im Verlauf des Algorithmus auftretende Zwischenwerte werden mit einem fortlaufenden Subindex durchnummeriert. Damit ergibt sich zum Beispiel für die Matrix U folgende Sequenz vom ursprünglichen zum aufdatierten Wert: $U \to U_1 \to U_2 \to \cdots \to U_+$.

4.5.1 Übergang zur symmetrischen Darstellung

Im ersten Schritt wird das allgemeine, rechteckige Problem durch die Multiplikationen $A_+A_+^T$ bzw. $A_+^TA_+$ in eine symmetrische Darstellungen überführt. In faktorisierter Form ergibt sich aus dem ersten Produkt zunächst die symmetrische Darstellung

$$\begin{aligned} A_+A_+^T &= U_+\Sigma_+ \overbrace{V_+^TV_+}^{I} \Sigma_+^TU_+^T \\ &= (U\Sigma V^T + uv^T)(V\Sigma U^T + vu^T) \\ &= U\Sigma\Sigma^TU^T + \tilde{v}u^T + u\tilde{v}^T + \gamma_1 uu^T, \end{aligned} \tag{4.40}$$

wobei $\tilde{v} = U\Sigma V^Tv \in \mathbb{R}^m$ und $\gamma_1 = v^Tv$ ist.

Weiterhin lassen sich die drei Rang-1 Terme zu einer Niedrigrangkorrektur vom Rang

zwei zusammenfassen. Damit reduziert sich (4.40) zu

$$\begin{aligned} U_+ \underbrace{\Sigma_+\Sigma_+^T}_{D_+} U_+^T &= U \underbrace{\Sigma\Sigma^T}_{D} U^T + \begin{bmatrix} u & \tilde{v} \end{bmatrix} \begin{bmatrix} \gamma_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u^T \\ \tilde{v} \end{bmatrix} \\ &= UDU^T + \begin{bmatrix} u & \tilde{v} \end{bmatrix} Q_1 \begin{bmatrix} \rho_1 & 0 \\ 0 & \rho_2 \end{bmatrix} Q_1^T \begin{bmatrix} u^T \\ \tilde{v}^T \end{bmatrix} \\ &= UDU^T + \rho_1 u_1 u_1^T + \rho_2 v_1 v_1^T. \end{aligned} \tag{4.41}$$

Hierbei ist

$$\begin{bmatrix} \gamma_1 & 1 \\ 1 & 0 \end{bmatrix} = Q_1 \begin{bmatrix} \rho_1 & 0 \\ 0 & \rho_2 \end{bmatrix} Q_1^T$$

eine Schurzerlegung mit $\rho_1 \geq 0$, $\rho_2 \leq 0$ und $D = \mathrm{diag}(\sigma_i^2)_{1 \leq i \leq m}$. Die Vektoren $u_1, v_1 \in \mathbb{R}^m$ werden durch

$$\begin{bmatrix} u_1 & v_1 \end{bmatrix} = \begin{bmatrix} u & \tilde{v} \end{bmatrix} Q_1$$

berechnet.

Aus (4.41) folgt, dass U_+ und D_+ durch die zwei aufeinander folgenden Rang-1 Aufdatierungen

$$U_1 D_1 U_1^T = UDU^T + \rho_1 u_1 u_1^T \tag{4.42}$$

$$U_+ D_+ U_+^T = U_1 D_1 U_1^T + \rho_2 v_1 v_1^T \tag{4.43}$$

berechnet werden können.

Im Gegensatz zu (4.40) führt die transponierte Multiplikation von A_+ zu der folgenden symmetrischen Darstellung

$$\begin{aligned} A_+^T A_+ &= V_+ \Sigma_+^T \overbrace{U_+^T U_+}^{I} \Sigma_+ V_+^T \\ &= (V\Sigma^T U^T + vu^T)(U\Sigma V^T + uv^T) \\ &= V\Sigma^T \Sigma V^T V^T + \tilde{u}v^T + v\tilde{u}^T + \gamma_2 vv^T, \end{aligned} \tag{4.44}$$

wobei $\tilde{u} = V\Sigma^T U^T u \in \mathbb{R}^n$ sowie $\gamma_2 = u^T u$ ist. Analog zu (4.40) lässt sich (4.44) zu einem

symmetrischen Rang-2 Aufdatierungsproblem der Matrizen V und Σ zusammenfassen:

$$\begin{aligned} V_+ \underbrace{\Sigma_+^T \Sigma_+}_{E_+} V_+^T &= V \underbrace{\Sigma^T \Sigma}_{E} V^T + \begin{bmatrix} v & \tilde{u} \end{bmatrix} \begin{bmatrix} \gamma_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v^T \\ \tilde{u} \end{bmatrix} \\ &= VEV^T + \begin{bmatrix} v & \tilde{u} \end{bmatrix} Q_2 \begin{bmatrix} \rho_3 & 0 \\ 0 & \rho_4 \end{bmatrix} Q_2^T \begin{bmatrix} v^T \\ \tilde{u} \end{bmatrix} \\ &= VEV^T + \rho_3 u_2 u_2^T + \rho_4 v_2 v_2^T \quad . \end{aligned} \tag{4.45}$$

Dabei sind

$$\begin{bmatrix} \gamma_2 & 1 \\ 1 & 0 \end{bmatrix} = Q_2 \begin{bmatrix} \rho_3 & 0 \\ 0 & \rho_4 \end{bmatrix} Q_2^T$$

mit $\rho_3 \geq 0$, $\rho_4 \leq 0$, und $u_2, v_2 \in \mathbb{R}^n$ durch

$$\begin{bmatrix} u_2 & v_2 \end{bmatrix} = \begin{bmatrix} v & \tilde{u} \end{bmatrix} Q_2$$

definiert. Weiterhin ist

$$E = \begin{bmatrix} \mathrm{diag}(\sigma_i^2)_{1 \leq i \leq m} & 0_{m \times d} \\ 0_{d \times m} & 0_{d \times d} \end{bmatrix}$$

mit $d = n - m$.

Aus (4.45) folgt, dass sich V_+ sowie E_+ durch die Aufdatierungssequenz

$$V_1^T E_1 V_1 = V^T E V + \rho_3 u_2 u_2^T \tag{4.46}$$

$$V_+^T E_+ V_+ = V_1^T E_1 V_1 + \rho_4 v_2 v_2^T \tag{4.47}$$

berechnen lassen.

Aufgrund des Zusammenhangs von E und D

$$E = \begin{bmatrix} D & 0_{m \times d} \\ 0_{d \times m} & 0_{d \times d} \end{bmatrix}, \tag{4.48}$$

können die Singulärwerte σ_i^+ sowohl durch (4.43) als auch durch (4.47) berechnet werden.

Mit Hilfe der symmetrischen Darstellungen (4.40) und (4.45) folgt für das ursprüngliche Problem (4.38), dass die Matrizen U_+, V_+ und Σ_+ mit Hilfe von zwei symmetrischen Rang-2 Aufdatierungen berechnet werden können:

$$U, D, V \overset{(4.42)(4.46)}{\longrightarrow} U_1, D_1, V_1 \overset{(4.43)(4.47)}{\longrightarrow} U_+, D_+, V_+.$$

Für jeden der beiden Schritte ist zunächst die Berechnung der Diagonalelemente von D_1 bzw. D_+ nötig, bevor die rechten- bzw. linken Singulärvektoren aufdatiert werden können. Das Vorgehen zur Lösung der einzelnen symmetrischen Probleme ist nachfolgend im Detail anhand der ersten Rang-1 Korrektur

$$U_1 D_1 U_1^T = U D U^T + \rho_1 u_1 u_1^T$$

beschrieben.

4.5.2 Aufdatierung der Singulärwerte

Die effiziente Berechnung der aufdatierten Matrix D_1 wird in Kapitel 4.2 ausführlich dargestellt. Dieser Vorgehensweise folgend, reduziert man (4.49) zunächst auf das diagonale Problem

$$\begin{aligned} U_1 D_1 U_1^T &= U D U^T + \rho_1 u_1 u_1^T \\ &= U \underbrace{(D + \rho_1 \tilde{u}_1 \tilde{u}_1^T)}_{B} U^T, \end{aligned} \tag{4.49}$$

wobei $\tilde{u}_1 = U u_1$ ist. Nach einer Schur-Zerlegung der Matrix B ergibt sich

$$C D_1 C^T = D + \rho_1 \tilde{u}_1 \tilde{u}_1^T \tag{4.50}$$

mit der orthogonalen Matrix $C \in \mathbb{R}^{m \times m}$. Mit Hilfe von (4.49) und (4.50) ist die Berechnung der Matrix D_1 mit $\mathcal{O}(m^2)$ Operationen möglich, vgl. Kapitel 4.2.

Bemerkung 4.5.1. *Dieses Vorgehen ist ein weiteres Mal zur Berechnung von D_+ aus D_1 nötig. Durch die Diagonalelemente der Matrix $D_+ = \operatorname{diag}(d_i^+)_{1 \le i \le m}$ sind die aufdatierten Singulärwerte durch*

$$\sigma_i^+ = \sqrt{d_i^+}, \quad 1 \le i \le m \tag{4.51}$$

eindeutig definiert, vgl. (4.48).

Durch Einsetzen von (4.50) in (4.49) folgt weiterhin

$$U_1 D_1 U_1^T = \underbrace{U(C}_{U_1} D_1 \underbrace{C^T)U^T}_{U_1^T}. \tag{4.52}$$

Offensichtlich ist damit die Aufdatierung der Matrix U_1 äquivalent zur Berechnung des Matrixproduktes UC. Im Folgenden wird zunächst die spezielle Struktur der Matrix C gezeigt. In Kapitel 4.3 wurden bereits verschiedene Möglichkeiten vorgestellt, wie durch Ausnutzen dieser Struktur eine effiziente Matrixmultiplikation möglich ist.

4.5.3 Berechnung der Cauchy-Matrix

Aufgrund der Orthogonalität von C lässt sich (4.50), wie im Folgenden dargestellt, umschreiben. Zur besseren Lesbarkeit soll an dieser Stelle die vereinfachte Schreibweise $u = \tilde{u}_1$ sowie $\rho = \rho_1$ verwendet werden:

$$\begin{aligned} CD_1C^T &= D + \rho uu^T \\ \Leftrightarrow DC - CD_1 &= -\rho uu^T C \quad . \end{aligned} \tag{4.53}$$

Durch die Lösung der Sylvestergleichung (4.53) ist die Matrix C eindeutig definiert.

Theorem 4.5.1. *Die Lösung $X \in \mathbb{R}^{n\times n}$ der diagonalen Sylvestergleichung*

$$\begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix} X - X \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_n \end{bmatrix} + M = 0 \tag{4.54}$$

mit $M \in \mathbb{R}^{n\times n}$ und paarweise verschiedenen $a_i \neq b_j$ für $i, j = 1, \ldots, n$ ist gegeben durch

$$X_{ij} = \frac{M_{ij}}{b_j - a_i}.$$

Ist M eine Matrix vom Rang eins mit der Darstellung $M = cd^T$, so ist die Lösung von (4.54) durch

$$X = \begin{bmatrix} c_1 & & \\ & \ddots & \\ & & c_n \end{bmatrix} \begin{bmatrix} (b_1 - a_1)^{-1} & \cdots & (b_n - a_1)^{-1} \\ \vdots & & \vdots \\ (b_1 - a_n)^{-1} & \cdots & (b_n - a_n)^{-1} \end{bmatrix} \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix} \tag{4.55}$$

explizit gegeben.

Beweis. Folgt direkt aus Einsetzen von (4.55) in (4.54). □

Wendet man obiges Theorem auf (4.53) an, folgt daraus

$$C = \rho \begin{bmatrix} u_1 & & \\ & \ddots & \\ & & u_n \end{bmatrix} \begin{bmatrix} (\lambda_1^+ - \lambda_1)^{-1} & \cdots & (\lambda_n^+ - \lambda_1)^{-1} \\ \vdots & & \vdots \\ (\lambda_1^+ - \lambda_n)^{-1} & \cdots & (\lambda_n^+ - \lambda_n)^{-1} \end{bmatrix} \begin{bmatrix} u^T c_1 & & \\ & \ddots & \\ & & u^T c_n \end{bmatrix}, \tag{4.56}$$

wobei $C = \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix}$ und $c_i \in \mathbb{R}^n$ die Spalten von C sind.

Aufgrund von (4.53) ist auch die rechte Seite der Darstellung (4.56) von C abhängig. Dennoch lässt sich aus der impliziten Form (4.56) eine explizite Darstellung für C ableiten. Bei spaltenweiser Betrachtung der Gleichung (4.56) lässt sich jede Spalte c_i durch das Eigenwertproblem

$$c_i = \rho \begin{bmatrix} \frac{u_1}{\lambda_i^+ - \lambda_1} \\ \vdots \\ \frac{u_n}{\lambda_i^+ - \lambda_n} \end{bmatrix} u^T c_i \tag{4.57}$$

definieren.

Die Lösung von (4.57) ist durch

$$c_i = \alpha_i \begin{bmatrix} \frac{u_1}{\lambda_1 - \lambda_i^+} & \cdots & \frac{u_n}{\lambda_n - \lambda_i^+} \end{bmatrix}^T \tag{4.58}$$

mit $\alpha_i \in \mathbb{R}$ gegeben. Dies lässt sich durch Einsetzen von (4.58) in (4.57) zeigen. Denn sei $c_i = \begin{bmatrix} c_{i,1} & \cdots & c_{i,j} & \cdots & c_{i,n} \end{bmatrix}^T$, dann gilt für jedes Element

$$\begin{aligned} c_{i,j} &= \alpha_i \frac{u_j}{\lambda_j - \lambda_i^+} \\ &= \alpha_i \rho \sum_{k=1}^{n} \frac{u_k^2 u_j}{(\lambda_j - \lambda_i^+)(\lambda_i^+ - \lambda_k)} \\ \Leftrightarrow 0 &= \alpha_i \underbrace{\left(1 + \rho \sum_{k=1}^{n} \frac{u_k^2}{\lambda_k - \lambda_i^+} \right)}_{=0} \quad . \end{aligned} \tag{4.59}$$

Der Klammerausdruck entspricht genau der zum Aufdatierungsproblem (4.50) gehörigen Säkulargleichung, vgl. (4.11). Demzufolge ist dieser Term gleich Null, da λ_k und λ_i^+ die Eigenwerte der Matrizen D bzw. D_+ sind. Aufgrund dessen kann α_i beliebig gewählt werden. Dies wird zur Normierung der Spalten von C genutzt, um die vorausgesetzte Orthogonalität der Matrix zu erhalten.

Zusammenfassend ergibt sich für die orthogonale Matrix C der Zerlegung (4.50) die folgende Darstellung:

$$C = \underbrace{\begin{bmatrix} u_1 & & \\ & \ddots & \\ & & u_n \end{bmatrix} \overbrace{\begin{bmatrix} \frac{1}{\lambda_1 - \lambda_1^+} & \cdots & \frac{1}{\lambda_1 - \lambda_n^+} \\ \vdots & & \vdots \\ \frac{1}{\lambda_n - \lambda_1^+} & \cdots & \frac{1}{\lambda_n - \lambda_n^+} \end{bmatrix}}^{\tilde{C}} \begin{bmatrix} \|\hat{c}_1\|_2 & & \\ & \ddots & \\ & & \|\hat{c}_n\|_2 \end{bmatrix}^{-1}}_{\hat{C} = \begin{bmatrix} \hat{c}_1 & \cdots & \hat{c}_n \end{bmatrix}} \quad . \tag{4.60}$$

Definition (4.60) zeigt, dass die Struktur von C ähnlich einer Cauchy-Matrix, vgl. Kapitel 2.6.1, ist. Insbesondere entspricht C der beidseitig diagonal skalierten Cauchy-Matrix $\tilde{C}$. Die per Definition geforderte Orthogonalität von C wird durch die freie Wahlmöglichkeit des Parameters α_i, vgl. Kapitel (4.59), welche sich in der rechten Diagonalskalierung widerspiegelt, sichergestellt.

4.5.4 Aufdatierung der Singulärvektoren

Die Aufdatierung der orthogonalen Matrix U aus (4.52) entspricht der Berechnung des Matrixprodukts $U_1 = UC$. Die besondere Struktur von C ermöglicht die Berechnung dieses Produkts mit einer geringeren Komplexität als $\mathcal{O}(n^3)$, welche im Fall allgemeiner, vollbesetzter Matrizen nötig wäre. Das Matrixprodukt $U_1 = UC$ kann in folgenden drei Schritten effizient berechnet werden:

(i) Berechnung von $U_{(i)} = U \cdot \begin{bmatrix} u_1 & & \\ & \ddots & \\ & & u_n \end{bmatrix}$.

(ii) Berechnung von $U_{(ii)} = U_{(i)} \cdot \begin{bmatrix} \frac{1}{\lambda_1 - \lambda_1^+} & \cdots & \frac{1}{\lambda_1 - \lambda_n^+} \\ \vdots & & \vdots \\ \frac{1}{\lambda_n - \lambda_1^+} & \cdots & \frac{1}{\lambda_n - \lambda_n^+} \end{bmatrix}$.

(iii) Berechnung von $U_1 = U_{(ii)} \cdot \begin{bmatrix} \|\hat{c}_1\|_2 & & \\ & \ddots & \\ & & \|\hat{c}_n\|_2 \end{bmatrix}$.

Die Produkte (i) und (iii) beinhalten jeweils eine vollbesetzte sowie eine Diagonalmatrix und sind demnach mit einer Komplexität von $\mathcal{O}(n^2)$ berechenbar. Zur effizienten Berechnung von Produkt (ii) kann die spezielle Struktur der Matrix $\tilde{C}$ ausgenutzt werden. Diese Multiplikation entspricht der n-fachen Multiplikation einer Cauchy-Matrix mit einem Vektor und ist im günstigsten Fall mit einem Gesamtaufwand von $\mathcal{O}(n^2 \log^2 n)$ möglich. Verschiedene Algorithmen zur dazu notwendigen Multiplikation von Cauchy-Matrizen mit Vektoren wurden bereits in Kapitel 4.3 vorgestellt.

Bemerkung 4.5.2. *Die Multiplikationen (i)-(iii) müssen für jede der vier symmetrischen Niedrigrangaufdatierungen (4.42), (4.43), (4.46) und (4.47) durchgeführt werden. Die hier verwendete Dimension n dient nur der Vereinfachung der Schreibweise und stellt*

den ungünstigsten zu erwartenden Fall dar. Unter Umständen sind Multiplikationen mit Matrizen von geringerer Dimension nötig. Diese reduziert sich zum Beispiel durch mehrfache Eigenwerte oder spezielle Strukturen in den Aufdatierungsvektoren. Siehe dazu auch Kapitel 4.2.1.

4.5.5 Korrekturschritt

In exakter Arithmetik gilt nach Durchführung der vier symmetrischen Rang-1 Aufdatierungen (4.42), (4.43), (4.46) und (4.47) die Gleichung

$$U\Sigma V^T + uv^T = UC_u\Sigma_+C_v^TV^T. \tag{4.61}$$

Hierbei ist $C_u = C_{u_2}C_{u_1}$ das Produkt der beiden Cauchymatrizen, welche während der Aufdatierungen von U durch (4.42), (4.43), und $C_v = C_{v_2}C_{v_1}$ das Produkt der beiden Cauchymatrizen, welche während der Aufdatierungen von V durch (4.46), (4.47) auftreten. Damit gilt weiterhin

$$C_u^T(\Sigma + \tilde{u}\tilde{v}^T)C_v = \Sigma_+, \tag{4.62}$$

mit $U\tilde{u} = u$ und $V\tilde{v} = v$.

Aufgrund der expliziten Verwendung der Produkte A^TA und AA^T sowie der separaten Aufdatierung der Matrizen U und V ergibt sich bei der Durchführung des bis hierher beschriebenen Algorithmus eine gestörte rechte Seite in (4.62). Dadurch ist

$$C_u^T(\Sigma + \tilde{u}\tilde{v}^T)C_v = \underbrace{\Sigma_+ + \varepsilon}_{E}, \tag{4.63}$$

wobei $\varepsilon \in \mathbb{R}^{m\times n}$ die Störung der rechten Seite darstellt. Die Matrix $E = (e_{i,j})_{1\le i\le n, 1\le j\le m}$ ist demzufolge eine fehlerbehaftete Annäherung an Σ_+.

Bemerkung 4.5.3. *Eine explizite Berechnung von E ist in effizienter Weise möglich, da die rechte Seite von (4.63) ausschließlich Cauchy-Matrizen sowie eine Diagonalmatrix enthält.*

Da $\|\varepsilon\|_F \ll \|\Sigma_+\|_F$ gilt, ist es möglich, E mit relativ geringem Berechnungsaufwand auf Diagonalform zu transformieren und somit Σ_+ zu erhalten. Ein dazu mögliches Vorgehen wird im Folgenden beschrieben.

Zunächst ist festzuhalten, dass Einträge der Größenordnung

$$\epsilon_E = n \cdot |e_{1,1}| \cdot \epsilon, \tag{4.64}$$

wobei ϵ die Maschinengenauigkeit bezeichnet, vernachlässigbar sind und somit als Null angenommen werden können. Zahlreiche numerische Beispiele haben ergeben, dass nur wenige Außendiagonalelemente von E betraglich größer als ϵ_E sind.

Beispiel 4.5.1. *Eine Veranschaulichung der typischen Größenordnung der Elemente in E wird durch Abbildung (4.1) gegeben. Diese zeigt die 250-te Zeile einer Matrix $E \in \mathbb{R}^{500\times 750}$. In diesem Beispiel ist $e_{1,1} = 640$. Daraus folgt, dass nach (4.64) sämtliche*

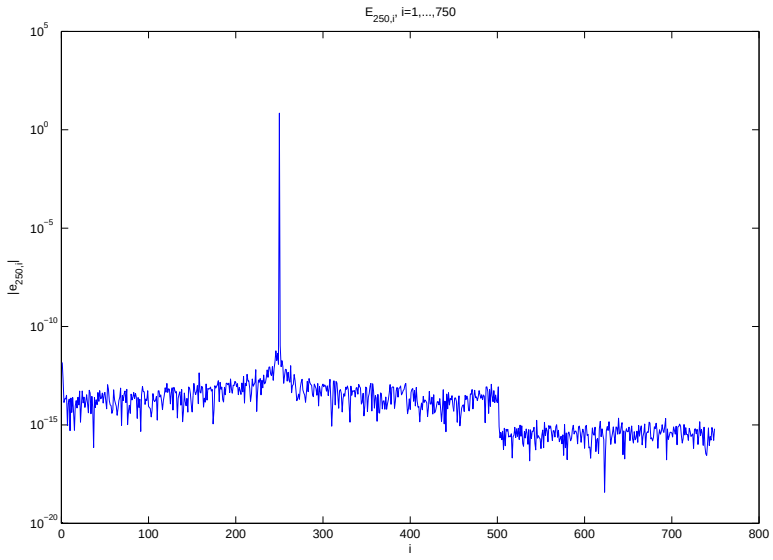


Abbildung 4.1: Elemente der 250-ten Zeile einer typischen Matrix E

Einträge mit Betrag kleiner 10^{-11} als Null angenommen werden können. Damit verbleiben nur noch wenige zu eliminierende Elemente in E. Diese befinden sind typischerweise nahe der Hauptdiagonalen.

Elimination der Außendiagonalelemente

Ziel ist es, die Außendiagonalelemente von E auf betragsmäßig kleinere Werte, als durch (4.64) definiert ist, zu reduzieren. Eine Möglichkeit dazu besteht darin, die symmetrische Matrix $S = E^T E$ zu nutzen, um mit Hilfe der Jacobi-Methode, vgl. Kapitel 2.1.3, bzw. dem symmetrischen QR-Verfahren, vgl. Kapitel 2.3, die Außendiagonalelemente von S zu eliminieren. Die so berechneten orthogonalen Transformationen werden dabei direkt auf E angewendet, um die explizite Berechnung von S zu vermeiden. Dies führt zu

Transformationen der Art

$$\underbrace{UC_uT_1}_{U_+}\underbrace{T_1^TET_2}_{\Sigma_+}\underbrace{T_2^TC_v^TV^T}_{V_+^T} = U_+\Sigma_+V_+^T,$$

wobei aufgrund der Orthogonalität von T_1 und T_2 auch die Orthogonalität von U und V erhalten bleibt.

Eine weitere Möglichkeit zur Elimination der Außendiagonaleinträge von E ist die Anwendung der beidseitigen Jacobi-Methode. Hierbei wird im Wesentlichen eine Folge von (2×2)-SVD Problemen gelöst.

Ein Schritt dieses Vorgehens kann folgendermaßen durchgeführt werden. Sei $\tilde{E} = \begin{bmatrix} e_{ii} & e_{ij} \\ e_{ji} & e_{jj} \end{bmatrix}$, mit $1 \leq i, j \leq m$ eine Teilmatrix von E, in welcher die Außendiagonaleinträge e_{ij} sowie e_{ji} eliminiert werden sollen. Dazu werden zwei Givensrotationen G_{ij} sowie H_{ij} so bestimmt, dass diese nach recht- bzw. linksseitiger Multiplikation mit $\tilde{E}$

$$\underbrace{\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}^T}_{G_{ij}^T}\begin{bmatrix} e_{ii} & e_{ij} \\ e_{ji} & e_{jj} \end{bmatrix}\underbrace{\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}}_{H_{ij}} = \begin{bmatrix} \tilde{e}_{ii} & 0 \\ 0 & \tilde{e}_{jj} \end{bmatrix} \tag{4.65}$$

eine Diagonalmatrix ergeben.

Typischerweise ist eine Sequenz dieser (2×2)-Subprobleme auf den tridiagonalen Anteil von E im hier vorliegenden Fall anzuwenden. Damit ergibt sich die Transformation

$$\underbrace{UC_uG_1}_{U_+}\underbrace{G_1^TEH_1}_{\Sigma_+}\underbrace{H_1^TC_v^TV^T}_{V_+^T} = U_+\Sigma_+V_+^T, \tag{4.66}$$

mit

$$\begin{aligned} G_1 &= G_{12}G_{23}\cdots G_{m-1,m} \\ H_1 &= H_{12}H_{23}\cdots H_{m-1,m}. \end{aligned}$$

Wie im Fall der einseitigen Jacobi-Methode werden dabei auch die weiteren Außendiagonalelemente von E betragsmäßig reduziert. Die Durchführung einer solchen Korrektursequenz (4.66) erfordert aufgrund der Multiplikationen $UC_u \cdot G_1$ sowie $VC_v \cdot H_1$ einen Rechenaufwand von $\mathcal{O}(n^2)$.

Bemerkung 4.5.4. *Ist $|e_{i,j}| > n \cdot |e_{1,1}| \cdot \epsilon$ mit $1 \leq i \leq m$ und $m < j \leq n$ ein Element im hinteren Teil von E, so kann dieses durch eine Givensrotation der Spalten i und j eliminiert werden.*

Korrektur der Vorzeichen

Wie bereits in Kapitel 2.3 beschrieben, sind die zu einem einfachen Singulärwert σ_i gehörigen rechten Singulärvektoren v_i nur bis auf den Betrag eindeutig. Der zugehörige linke Singulärvektor u_i ist danach durch

$$Av_i = \sigma_i u_i$$

eindeutig definiert. Durch die separate Berechnung der neuen Matrizen U_+ und V_+, welche die Singulärvektoren von A_+ enthalten, ist es möglich, dass nach erfolgter Aufdatierung die Vorzeichen einzelner rechter- und linker Singulärvektoren nicht zusammenpassen, d.h. es ist $A_+ v_i^+ = -\sigma_i^+ u_i^+$. In einem solchen Fall müssen entweder die Vorzeichen des Vektors u_i^+ oder v_i^+ gewechselt werden. Ein solches Vorgehen ist für alle Paare u_i^+, v_i^+ nötig, bei denen der zugehörige Diagonaleintrag e_{ii} der Matrix E negativ ist. Um den korrekten, zugehörigen Singulärwert σ_i^+ zu erhalten, muss e_{ii} ebenfalls mit -1 multipliziert werden.

Ein ähnliches Problem kann bei Singulärwerten σ_j mit einer Vielfachheit größer eins auftreten. Durch die separate Aufdatierung von U und V ist es möglich, dass die durch die zugehörigen linken- bzw. rechten Singulärvektoren aufgespannten Räume nicht zusammenpassen. Tritt ein solcher Fall ein, existieren in E Außendiagonalelemente, welche betragsmäßig von ähnlicher Größenordnung wie die Diagonaleinträge der entsprechenden Zeile sind. Diese können durch die Anwendung von Givensrotationen gemäß (4.65) eliminiert werden. Ein Beispiel dazu ist im Kapitel 4.6 gegeben.

4.5.6 Zusammenfassung des SVD Aufdatierungsalgorithmus

Der neu entwickelte SVD-Aufdatierungsalgorithmus zur Lösung des in Abschnitt 4.4 beschriebenen Problems ist in den Kapiteln 4.5.1 - 4.5.5 vollständig beschrieben. Zusammenfassend besteht dieser aus folgenden Schritten:

- Zunächst wird das unsymmetrische Rang-1 Aufdatierungsproblem der Matrizen U,V und Σ in je zwei symmetrische Probleme vom Rang zwei überführt. In jedem dieser Rang-2 Probleme tritt nur eine der beiden orthogonalen Matrizen U oder V auf.

- Jedes der beiden Rang-2 Probleme wird als Folge zweier Rang-1 Aufdatierungen behandelt. Diese werden wenn möglich zunächst in der Dimension reduziert. Darauf folgend können jeweils mit Hilfe der Säkulargleichung die Eigenwerte aufdatiert

werden. Die aufdatierten Singulärwerte von A_+ sind durch die Quadratwurzel dieser definiert.

- Sind die aufdatierten Eigenwerte bekannt, wird durch diese eine beidseitig skalierte Cauchy-Matrix $C_{(t,s)}$ definiert. Deren Struktur wird ausgenutzt, um durch effiziente Matrixmultiplikationen die aufdatierten orthogonalen Matrizen U_+ bzw. V_+ zu berechnen.

- Aufgrund der separaten Berechnung von U_+ und V_+ sowie wegen eventuell auftretender numerischer Probleme aufgrund der expliziten Verwendung der Matrixprodukte A^TA und AA^T ist ein abschließender Korrekturschritt nötig, um die gewünschte Genauigkeit der Singulärwertzerlegung von A_+ zu erhalten.

Jeder dieser Schritte ist in deutlich weniger als $\mathcal{O}(n^3)$ Operationen durchführbar.

4.6 Numerische Ergebnisse

Der hier vorgestellte Algorithmus zur Aufdatierung der Singulärwertzerlegung wurde in MATLAB[1] implementiert und hinsichtlich seiner Genauigkeit sowie Geschwindigkeit an zahlreichen Beispielen getestet. Einige der dabei entstandenen Ergebnisse werden im Folgenden wiedergegeben. Diese wurden auf einer AMD Athlon 64 X2 4400+ Maschine erzielt. Dabei ist festzuhalten, dass bei Laufzeitvergleichen teilweise selbst entwickelter MATLAB Code mit MATLAB Binaries verglichen wird, was sich als nachteilhaft für Ersteren auswirkt.

Matrix Multiplikation

Zunächst sollen die Ergebnisse eines Laufzeitvergleich sämtlicher Multiplikationen von Cauchy-Matrix mal Matrix, welche während der gesamten SVD Aufdatierung auftreten, gegeben werden. Hierbei wurden die Multiplikationstechniken, welche auf einer approximierten Cauchy-Matrix basieren, miteinander verglichen. Die Multiplikation mit Hilfe des erweiterten GGS-Algorithmus, vgl. Kapitel 4.3.1, hat sich nur für Probleme kleiner Dimension als numerisch stabil erwiesen und ist somit für einen Laufzeitvergleich irrelevant. Zur Approximation der Cauchy-Matrix wurden zum einen exponentielle Summen und zum anderen polynomielle Summen, vgl. Kapitel 4.3.3 verwendet. Dabei wurde eine Approximationsgenauigkeit von $\epsilon \leq 10^{-15}$ gefordert. Um diesen Wert zu erreichen, ist im

[1] MATLAB version 7.10.0. Natick, Massachusetts: The MathWorks Inc., 2003

Fall der polynomiellen Summen ein Rang von 32 und im Fall von exponentiellen Summen ein Rang von 10 nötig. Aus diesem Grund wurden bei der Verwendung der polynomiellen Summen nur diejenigen zulässigen Blöcke der Cauchy-Matrix durch Niedrigrang approximiert, deren Dimension größer (40×40) war. Für die Approximation durch exponentiellen Summen wurde eine minimale Blockgröße von (20×20) gefordert. In der

Dimension	$n > 20$	poly. Summen	$n > 40$	exp. Summen
750	228	3.467	55	1.701
1500	571	17.643	221	10.162

Abbildung 4.2: Matrixmultiplikation

zweiten und vierten Spalte der Tabelle 4.2 ist die Anzahl der Blöcke der Cauchy-Matrix gegeben, welche unter Beachtung der Zulässigkeitsbedingung (4.22) jeweils eine solche minimal geforderte Dimension aufwiesen. Die weiteren Spalten zeigen die Laufzeit sämtlicher durchgeführter Matrixmultiplikationen für beide Möglichkeiten der Approximation in Sekunden.

Einzelne Rang-1 Aufdatierungen

Im nächsten Beispiel sollen einige numerische Resultate einzelner Rang-1 Aufdatierungen für verschiedene Matrizen dargestellt werden. Hierbei wurden zum einen mit der MATLAB Routine random zufällig erzeugte, sowie zwei Matrizen des Matrix-Market [60] mit zufällig generierten Rang-1 Termen aufdatiert. Während der Berechnung wurden zwei Eigen- bzw. Singulärwerte als identisch festgelegt, wenn deren Differenz $|\tau| \leq 10^{-10}$ war. Die Ergebnisse dieser Aufdatierungen sind in Tabelle 4.3 dargestellt. Neben Dimension und Art der Matrix werden in der dritten und vierten Spalte zunächst die aufdatierten Matrizen U_+ bzw. V_+ auf ihre Orthogonalität untersucht. Dies geschieht durch die Differenzen $\|U_+\|_2 - 1$ sowie $\|V_+\|_2 - 1$, welche im exakt orthogonalen Fall gleich Null ergeben müssen. In den folgenden beiden Spalten sind die relativen Fehler der aufdatierten Singulärwerte ('Fehler σ')

$$\max\left(\frac{|\sigma_+ - \sigma_m|}{\max(\sigma_m)}\right) \tag{4.67}$$

sowie der Norm der aufdatierten Singulärwertzerlegung ('Fehler A_+')

$$\max\left(\frac{\|A_+ - U_+\Sigma_+V_+^T\|}{\max\sigma_m}\right)$$

Dimension	Typ	$\|U_+\| - 1$	$\|V_+\| - 1$	Fehler σ	Fehler A_+	KS
1000×1250	random	$1.5 \cdot 10^{-11}$	$7.9 \cdot 10^{-11}$	$2.2 \cdot 10^{-15}$	$8.8 \cdot 10^{-14}$	1
500×625	random	$1.8 \cdot 10^{-11}$	$2.0 \cdot 10^{-11}$	$2.9 \cdot 10^{-16}$	$4.3 \cdot 10^{-14}$	1
250×320	random	$3.7 \cdot 10^{-11}$	$1.8 \cdot 10^{-11}$	$5.8 \cdot 10^{-16}$	$9 \cdot 10^{-14}$	1
1500×1500	random	$2.1 \cdot 10^{-11}$	$3.5 \cdot 10^{-10}$	$1.6 \cdot 10^{-15}$	$1.1 \cdot 10^{-13}$	1
200×1500	random	$7.6 \cdot 10^{-12}$	$3.0 \cdot 10^{-12}$	$2.3 \cdot 10^{-15}$	$4.8 \cdot 10^{-14}$	1
497×506	beacxc	$3.4 \cdot 10^{-12}$	$1.4 \cdot 10^{-12}$	$4.4 \cdot 10^{-13}$	$4.3 \cdot 10^{-12}$	3
66×66	bcsstk02	$4.4 \cdot 10^{-14}$	$1.9 \cdot 10^{-10}$	$3.5 \cdot 10^{-15}$	$1.6 \cdot 10^{-11}$	2

Abbildung 4.3: Genauigkeit der aufdatierten Matrizen für verschiedene Beispiele

gegeben. Hierbei wurden die exakten Singulärwerte σ_m mit der MATLAB Routine svd bestimmt. In der letzten Spalte ist die Anzahl der durchgeführten Korrektursequenzen (KS), vgl. Kapitel 4.5.5, gegeben, welche nötig war, um die Nebendiagonalelemente der Matrix E vom Betrag gemäß (4.64) zu reduzieren.

Es ist zu erkennen, dass die Orthogonalität der Matrizen $\|U_+\|$ und $\|V_+\|$ im Rahmen der Toleranz τ gewährleistet ist. Des Weiteren ist festzuhalten, dass in jedem der Beispiele maximal zwei Korrektursequenzen erforderlich waren, um eine sehr gute Approximation der aufdatierten Singulärwerte zu erhalten. Dadurch bleibt der Gesamtaufwand der Aufdatierung deutlich unter $\mathcal{O}(n^3)$.

Dimension	Neuberechnung	Aufdatierung
1000×1250	39.502	30.728
500×625	5.642	5.237
250×320	0.659	1.354
1500×1500	95.941	62.335
200×1500	2.432	35.989
497×506	1.884	6.590
66×66	0.018	0.429

Abbildung 4.4: Laufzeit zur Aufdatierung verschiedener Singulärwertzerlegungen

In der zweiten Tabelle 4.4 sind die Laufzeiten zur Berechnung der Singulärwertzerlegung von A_+ für zufällig generierte Matrizen verschiedener Größe gegeben. Hierbei wird die Zeit, welche das in dieser Arbeit vorgestellte Verfahren im Vergleich zur Neube-

rechnung der SVD aus der zuvor explizit aufdatierten Matrix $A_+ = A + uv^T$ benötigt, dargestellt. Die Neuberechnung fand dabei mit der MATLAB Routine svd statt. Die Probleme wurden so gewählt, dass keine Reduktion der Dimension während der Aufdatierung möglich war. Obwohl in diesem Fall die Performance von selbst geschriebenem MATLAB Code mit MATLAB Binaries verglichen wird, ist zu sehen, dass der Algorithmus bei Problemen großer Dimension einen Laufzeitvorteil erbringt.

Aufdatierungssequenz

Im Folgenden wird der hier beschriebene Algorithmus verwendet, um eine Niedrigrangzerlegung bzw. Aufdatierung als Folge von Rang-1 Korrekturen zu berechnen. Dabei wird folgende Aufdatierungssequenz durchgeführt. Zunächst werden die Matrizen $A_0 \in \mathbb{R}^{n\times m}$ als Nullmatrix sowie $B_0 \in \mathbb{R}^{n\times m}$ zufällig initialisiert, damit ist auch die SVD $A_0 = U_0\Sigma_0 V_o^T$ bereits trivial gegeben. Im Weiteren wird in jedem Schritt k das betragsmäßig größte Element $b_{i,j}$ der Matrix B_k gesucht. Daraufhin wird das Schurkomplement von B_k berechnet

$$B_{k+1} = B_k - \frac{1}{b_{ij}^k} B_k e_i e_j^T B_k$$

und der darin auftretende Rang-1 Term zu A_k addiert

$$A_{k+1} = A_k + \frac{1}{b_{ij}^k} B_k e_i e_j^T B_k.$$

Diese Addition des Rang-1 Terms erfolgt jeweils direkt auf die Singulärwertzerlegung $A_k = U_k\Sigma_k V_k^T$. Bei exakter Durchführung dieses Vorgehens ist nach n Schritten $A_n = U_n\Sigma_n V_n^T = B_0$ sowie $B_n = 0$. Tabelle 4.5 zeigt die erzielte Genauigkeit einiger Zwischen- sowie die des Endwertes der aufdatierten Matrixfaktoren U_k, Σ_k sowie V_k für zwei Probleme der Dimesion 50×60 sowie 500×750. Hierbei ist durch die Spalten $\|U\|_2 - 1$ und $\|V\|_2 - 1$ analog zu Tabelle 4.3 ein Maß der Orthogonalität von U_k und V_k gegeben. Die jeweils letzte Spalte zeigt den relativen Fehler der aufdatierten Singulärwerte, welcher wie in 4.67 bestimmt wird. Die Werte $\max \left| \frac{B_0 - U_n\Sigma_n V_n^T}{\max \sigma_m} \right|$ zeigen den relativen Fehler des Endergebnisses. Hierbei ist $\max(\sigma_m)$ der größte, mit MATLAB exakt berechnete Singulärwert von B_0.

Spezialfall

Im letzten Abschnitt wird ein Beispiel dargestellt, in welchem die Aufdatierungsvektoren in 4.41 sowie 4.45 linear abhängig sind. Durch die daraus folgende, vollständig separate

50×60				500×750			
k	$\|U\|_2 - 1$	$\|V\|_2 - 1$	Fehler σ	k	$\|U\|_2 - 1$	$\|V\|_2 - 1$	Fehler σ
10	$2.0 \cdot 10^{-15}$	$6.1 \cdot 10^{-15}$	$2.1 \cdot 10^{-15}$	100	$5.5 \cdot 10^{-13}$	$5.9 \cdot 10^{-13}$	$1.2 \cdot 10^{-15}$
20	$3.5 \cdot 10^{-14}$	$6.4 \cdot 10^{-13}$	$3.0 \cdot 10^{-15}$	200	$3.0 \cdot 10^{-12}$	$5.5 \cdot 10^{-12}$	$1.3 \cdot 10^{-14}$
30	$1.8 \cdot 10^{-13}$	$4.6 \cdot 10^{-13}$	$3.0 \cdot 10^{-15}$	300	$7.6 \cdot 10^{-12}$	$6.3 \cdot 10^{-12}$	$9.6 \cdot 10^{-14}$
40	$3.4 \cdot 10^{-13}$	$4.0 \cdot 10^{-13}$	$1.9 \cdot 10^{-13}$	400	$4.6 \cdot 10^{-11}$	$8.9 \cdot 10^{-11}$	$3.8 \cdot 10^{-13}$
50	$4.0 \cdot 10^{-13}$	$3.3 \cdot 10^{-13}$	$4.9 \cdot 10^{-13}$	500	$8.7 \cdot 10^{-11}$	$9.1 \cdot 10^{-11}$	$5.6 \cdot 10^{-13}$
$\max\left(\frac{\|B_0 - U_n \Sigma_n V_n^T\|}{\max \sigma_m}\right) = 4.1 \cdot 10^{-13}$				$\max\left(\frac{\|B_0 - U_n \Sigma_n V_n^T\|}{\max \sigma_m}\right) = 6.3 \cdot 10^{-11}$			

Abbildung 4.5: Genauigkeit bei Durchführung einer Folge von Aufdatierungen

Aufdatierung der Matrizen U und V wird die Gleichung 4.62 verletzt und es folgt das zunächst $A_+ \neq U_+ \Sigma_+ V_+^T$ ist. Erst die Durchführung eines Korrekturschritts, wie in Kapitel 4.5.5 beschrieben, liefert die exakte Lösung. Die Werte im folgenden Beispiel sind zur besseren Übersicht auf drei Nachkommastellen gerundet.

Nach dem Übergang vom allgemeinen Aufdatierungsproblem

$$U_+ \Sigma_+ V_+^T = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{V^T} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

zur symmetrischen Darstellung, folgt zur Aufdatierung der Matrix U die Rang-2 Darstellung

$$U_+ \Sigma_+ \Sigma_+^T U_+^T = U \Sigma \Sigma^T U^T - 0.192 \begin{bmatrix} -0.792 \\ -0.792 \\ -0.792 \\ -0.792 \end{bmatrix} \begin{bmatrix} -0.792 \\ -0.792 \\ -0.792 \\ -0.792 \end{bmatrix}^T + 5.192 \begin{bmatrix} -1.171 \\ -1.171 \\ -1.171 \\ -1.171 \end{bmatrix} \begin{bmatrix} -1.171 \\ -1.171 \\ -1.171 \\ -1.171 \end{bmatrix}^T . \tag{4.68}$$

Offensichtlich sind beide Rang-1 Terme in (4.68) linear abhängig, womit diese zu einem einzelnen Rang-1 Update für U zusammengefasst werden können. Für die Aufdatierung der Matrix V trifft dies in analoger Weise zu. Aufgrund der separaten Aufdatierung von

U und V ergibt sich in diesem Fall die fehlerbehaftete Matrix Σ_+:

$$E = \begin{bmatrix} 5.385 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.894 & 0.447 & 0 \\ 0 & 0 & -0.447 & -0.894 & 0 \end{bmatrix}.$$

Nach Anwendung der unsymmetrischen Jacobi Methode auf die dritte und vierte Zeile bzw. Spalte erhält man schließlich das korrekte Endergebnis.

Dieses Beispiel zeigt, dass der in Kapitel 4.5.5 beschrieben Korrekturschritt für den hier vorgestellten Aufdatierungsalgorithmus unerlässlich zum Erhalt einer korrekten Lösung ist.

4.7 Zusammenfassung

In diesem Kapitel wurde ein neu entwickeltes Verfahren zur Rang-1 Aufdatierung der Singulärwertzerlegung für allgemeine, rechteckige Matrizen vorgestellt. Dieses ermöglicht die Aufdatierung der einzelnen Matrixfaktoren der Zerlegung mit einem deutlich geringeren Berechnungsaufwand, als er bei einer vollständigen Neufaktorisierung nötig wäre.

Zunächst wird das rechteckige Problem in zwei symmetrische Probleme überführt. Es stellt sich heraus, dass für jedes dieser Probleme eine Rang-2 Korrektur durchzuführen ist. Die Berechnung der aufdatierten Singulärwerte erfolgt unter Zuhilfenahme der Säkulargleichung. Hieraus folgt, dass die Aufdatierung der Singulärvektoren jeweils einer Matrix-Matrix Multiplikation entspricht und somit den rechenintensivsten Teil der Rang-1 Modifikation darstellt. Es konnte jedoch gezeigt werden, dass jeweils eine der zu multiplizierenden Matrizen die Struktur einer beidseitig diagonalskalierten Cauchy-Matrix besitzt. Unter Ausnutzung dieser Struktur kann der Rechenaufwand zur Matrixmultiplikation deutlich reduziert werden. Hierzu wurde zunächst der erweiterte GGS-Algorithmus in Kombination mit der Möglichkeit der Transformation von Cauchy-Matrizen vorgestellt. Dies ermöglicht eine Matrix-Matrix Multiplikation in einer Anzahl von $\mathcal{O}(n^2 \log^2 n)$ Operationen. Allerdings stellt sich heraus, dass dieses Vorgehen numerische Stabilitätsprobleme mit sich bringt. Aus diesem Grund wurde im Weiteren dazu übergegangen, die Cauchy-Matrix mit Hilfe hierarchischer Techniken zu approximieren. Hierzu wurden zwei verschiedene Möglichkeiten, welche zum einen auf einer Approximation durch polynomielle- und zum anderen auf Exponentialsummen beruhen, vorgestellt. Letztere ermöglicht eine effizientere Niedrigrangdarstellung von Matrixblöcken großer Di-

mension der Cauchy-Matrix. Mit Hilfe dieser Approximation ist es möglich die Matrix-Matrix Multiplikation in deutlich weniger als $\mathcal{O}(n^3)$ Operationen stabil zu berechnen.

In einem abschließenden Korrekturschritt des Verfahrens wird die Genauigkeit der Lösung verbessert. Dies ist nötig, da nach dem Übergang zu den symmetrischen Problemen zur Berechnung der aufdatierten Singulärwerte die Eigenwerte der Matrizen A^TA bzw. AA^T explizit genutzt werden.

Das hier vorgestellte Verfahren wurde in MATLAB implementiert und an zahlreichen Beispielen getestet.

Dieser neu entwickelte Algorithmus ist zur Lösung großer, dichtbesetzter Probleme konzipiert. Aus diesem Grund wurden eventuell auftretende Dünnbesetztheitsstrukturen nicht betrachtet. Ebenso wurde auf eine Parallelisierung der Algorithmen verzichtet.

Kapitel 5

Aufdatierung der Matrixexponentialfunktion

5.1 Motivation

Die Matrixexponentialfunktion ist die mit Abstand meist untersuchte und am häufigsten verwendete Matrixfunktion. Ihre Bedeutung ist unter anderem durch Differentialgleichungen der Art

$$\frac{dy}{dt} = Ay, \quad y(0) = c, \quad y \in \mathbb{C}^n,\ A \in \mathbb{C}^{n\times n}$$

begründet, deren Lösung durch

$$y(t) = e^{At}c$$

charakterisiert ist, siehe [4]. Differentialgleichungen dieser Gestalt treten neben zahlreichen anderen Anwendungen beispielsweise in der Kernspinresonanzspektroskopie [44] auf. Die Matrixexponentialfunktion spielt weiterhin eine Rolle in der Analyse von Netzwerken [19], für Markov Modelle [7] sowie in der Kontrolltheorie dynamischer Systeme [55].

Die Berechnung der Matrixexponentialfunktion erfordert einen hohen numerischen Berechnungsaufwand. Der hierfür hauptsächlich verwendete Algorithmus ist das Scaling und Squaring Verfahren (Skalieren und Quadrieren). Dieses ermöglicht die Berechnung von e^A in $\mathcal{O}(n^3)$ Operationen, wobei die hierin auftretende Konstante aufgrund zahlreicher durchzuführender Matrixmultiplikationen ≥ 6 ist. Andere Möglichkeiten zur Berechnung sind unter anderem durch die Reihendarstellung der Exponentialfunktion, durch die Darstellung von e^A als Cauchy-Integral oder durch die Verwendung der Schur- bzw. Jordan Form gegeben. Siehe hierzu auch Kapitel 2.7. In der praktischen Anwendung führen

diese zuletzt genannten Methoden allerdings meist zu einer unverhältnismäßig hohen Berechnungskomplexität bzw. führen auf numerische Stabilitätsprobleme, vgl. [59].

In dieser Arbeit ist die Berechnung der aufdatierten Matrixenxponentialfunktion von Interesse. Hierbei wird angenommen, dass das matrixwertige Argument A einer Rang-1 Korrektur unterliegt. Von besonderem Interesse ist dabei, inwieweit durch die Kenntnis der ursprünglichen Matrixexponentialfunktion e^A der Rechenaufwand zur Bestimmung des aufdatierten Wertes e^{A+ab^T} reduziert werden kann. Hierzu werden im Folgenden zwei Verfahren, welche zum einen auf dem Scaling und Squaring Algorithmus und zum anderen auf der Darstellung als Cauchy-Integral beruhen, vorgestellt. Die erzielten theoretischen Erkenntnisse werden zusätzlich durch numerische Beispiele belegt.

5.2 Problemdefinition

Gegeben sei die Matrixexponentialfunktion

$$E = e^A \quad \in \mathbb{C}^{n\times n} \tag{5.1}$$

der Matrix $A \in \mathbb{C}^{n\times n}$. Weiterhin seien $a \in \mathbb{C}^n$ und $b \in \mathbb{C}^n$ zwei beliebige Vektoren, welche eine Rang-1 Modifikation

$$A_+ = A + ab^T$$

der Matrix A darstellen.

Gesucht ist die aufdatierte Matrixexponentialfunktion

$$\begin{aligned} E_+ &= e^{A_+} \\ &= e^{A+ab^T}. \end{aligned} \tag{5.2}$$

Aufgrund der hohen Komplexität die für die direkte Berechnung von E_+ in (5.2) nötig ist, soll ein effizienter Algorithmus entwickelt werden, welcher die Kenntnis von E ausnutzt, um die aufdatierte Matrixexponentialfunktion als Summe

$$E_+ = E + K \tag{5.3}$$

mit $K \in \mathbb{C}^{n\times n}$ zu bestimmen.

Als weitere Frage stellt sich, ob aufgrund der Rang-1 Differenz von A und A_+ eine Aussage bezüglich der Struktur von K zu treffen ist.

In den folgenden Kapiteln werden zwei Verfahren zur Berechnung der Matrixexponentialfunktion vorgestellt und auf die hier beschriebene Niedrigrangaufdatierung angewendet.

Bemerkung 5.2.1. *Da die im skalaren Fall gültige Gleichung*

$$e^{A+B} = e^A e^B \tag{5.4}$$

bei matrixwertigen Exponenten nur gilt, wenn A und B kommutierende Matrizen sind, werden auf (5.4) basierende Aufdatierungsverfahren in dieser Arbeit nicht betrachtet.

5.3 Aufdatierung mittels Cauchy-Integral

Wie bereits in Kapitel 2.5 beschrieben wurde, ist die Matrixfunktion $f(A)$ durch das Cauchy-Integral wie folgt definiert:

$$f(A) = \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1}\,\mathrm{dz}\,. \tag{5.5}$$

Hierbei ist Γ eine einfach geschlossene Kurve, welche das Spektrum $\sigma(A)$ der Matrix A umschließt.

Diese Darstellung lässt sich folgendermaßen auf den Fall der Rang-1 Aufdatierung von A anwenden. Aus $A_+ = A + ab^T$ folgt

$$\begin{aligned}
f(A_+) &= \frac{1}{2\pi i} \int_\Gamma f(z)(zI - (A + ab^T))^{-1}\,\mathrm{dz} \\
&\overset{Kap.2.4}{=} \frac{1}{2\pi i} \int_\Gamma f(z) \left((zI - A)^{-1} + \frac{(zI - A)^{-1} ab^T (zI - A)^{-1}}{1 - b^T (zI - A)^{-1} a} \right) \mathrm{dz} \\
&= \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} dz + \frac{1}{2\pi i} \int_\Gamma f(z) \frac{(zI - A)^{-1} ab^T (zI - A)^{-1}}{1 - b^T (zI - A)^{-1} a}\,\mathrm{dz} \\
&= f(A) + \underbrace{\frac{1}{2\pi i} \int_\Gamma f(z) \frac{(zI - A)^{-1} ab^T (zI - A)^{-1}}{1 - b^T (zI - A)^{-1} a}\,\mathrm{dz}}_{D(z)},
\end{aligned} \tag{5.6}$$

wobei die Kurve Γ so gewählt sein muss, dass diese die Spektren $\sigma(A)$ sowie $\sigma(A + ab^T)$ umschließt. Aus (5.6) folgt, dass die Differenz zwischen altem und aufdatiertem Funktionswert durch den zweiten Summanden $D(z)$ definiert ist

$$f(A + ab^T) - f(A) = D(z).$$

Der Integrand $d(z)$ von $D(z)$ kann wie folgt umgeschrieben werden:

$$\begin{aligned}
d(z) &= f(z) \frac{(zI - A)^{-1} ab^T (zI - A)^{-1}}{1 - b^T (zI - A)^{-1} a} \\
&= f(z)(zI - A)^{-1} a (1 + b^T (zI - A - ab^T)^{-1} a) b^T (zI - A)^{-1} \\
&= f(z)(1 + b^T (zI - A - ab^T)^{-1} a)(zI - A)^{-1} ab^T (zI - A)^{-1}.
\end{aligned} \tag{5.7}$$

Die Verwendung dieser Darstellung wird für die folgenden theoretischen Betrachtungen von Vorteil sein.

Bemerkung 5.3.1. *Analog zu (5.6) können auch Aufdatierungen von höherem Rang dargestellt werden. Ein Beispiel dazu ist in Abschnitt 5.6.3 gegeben.*

5.3.1 Numerische Integration

Zur numerischen Berechnung des Integrals aus (5.6) wird zunächst die summierte Trapezregel betrachtet. Diese ist wie folgt definiert, vgl. [69].

Definition 5.3.1. *Zu $f \in C([a,b],\mathbb{R})$ wird durch*

$$T_h(f) := \sum_{i=0}^{N-1} \frac{h}{2}\left(f(x_i) + f(x_{i+1})\right)$$

mit $h = \frac{b-a}{N}$ und $x_i = a + ih$ die Trapezsumme von f mit Schrittweite h bezeichnet.

Theorem 5.3.1. *Sei $f \in C^2([a,b],\mathbb{R})$. Dann wird das Integral*

$$I(f) = \int_a^b f(z)\,\mathrm{dz}$$

auf dem Intervall $[a,b]$ durch $T_h(f)$ mit einem Fehler von

$$|I(f) - T_h(f)| \leq \frac{h^2(b-a)}{12} f^{(2)}(\xi), \quad \xi \in (a,b) \tag{5.8}$$

approximiert.

Beweis. Siehe [69]. □

Aus (5.8) ist ersichtlich, dass der Fehler der summierten Trapezregel im allgemeinen Fall mit der Ordnung h^2 gegen Null geht. Aufgrund der Eigenschaften des hier betrachteten Integrals $D(z)$ wird allerdings eine höhere Approximationsordnung erreicht.

Für die folgenden Betrachtungen sei angenommen, dass als Integrationsweg Γ in (5.6) ein Kreis

$$\Gamma : [0,2\pi] \to \mathbb{C}, t \to m_k + r_k e^{it}$$

mit Mittelpunkt m_k und Radius r_k gewählt wird. Dieser sei so bestimmt, dass er das Spektrum von A sowie von $A + ab^T$ umschließt. Sei zur besseren Lesbarkeit weiterhin $m_k = 0$. Dann ist

$$\tilde{D}(t) = \frac{r_k}{2\pi} \int_0^{2\pi} f(r_k e^{it}) e^{it} (r_k e^{it} I - A)^{-1} a (1 + b^T (r_k e^{it} I - (A + ab^T))^{-1} a) b^T (r_k e^{it} I - A)^{-1}\,\mathrm{dt}\,. \tag{5.9}$$

Aus (5.9) ergibt sich hierzu die approximierende Trapezsumme

$$\begin{aligned} T_h(\tilde{d}(t)) &= \frac{r_k}{2\pi} h \sum_{j=0}^{N-1} f(r_k e^{it_j}) e^{it_j} B^{-1} a (1 + b^T B_+^{-1} a) b^T B^{-1} \\ &= \frac{r_k}{2\pi} h \sum_{j=0}^{N-1} \underbrace{f(r_k e^{it_j}) e^{it_j} (1 + b^T B_+^{-1} a)}_{\alpha_j} \underbrace{B^{-1} a}_{\tilde{a}_j} \underbrace{b^T B^{-1}}_{\tilde{b}_j^T} \end{aligned} \tag{5.10}$$

mit $B = (r_k e^{it_j} I - A)$, $B_+ = (r_k e^{it_j} I - (A + ab^T))$, $t_j = jh$ und $h = \frac{2\pi}{N}$.

Diese Darstellung zeigt, dass (5.9) durch eine Summe von Rang-1 Termen approximiert werden kann. Der maximal auftretende Rang von $T_h(\tilde{d}(t))$ ist dabei davon abhängig, wieviele Intervalle N nötig sind, um die geforderte Approximationsgenauigkeit zu erhalten. Das heißt, je besser die Approximationsordnung des Verfahrens ist, desto geringer ist der Rang der Niedrigrangkorrektur K in (5.3). In Kapitel 5.6 wird anhand von Beispielen gezeigt, dass der numerische Rang deutlich geringer als N ist.

Für 2π-periodische Funktionen gelten für die summierte Trapezregel stärkere Fehlerschranken als im allgemeinen Fall (5.8). Diese sind für verschiedene Klassen von Funktionen unter anderem in [62, 74] dargestellt. Für das hier vorliegende Integral gilt in diesem Zusammenhang folgendes Theorem aus [74]:

Theorem 5.3.2. *Sei $f : \mathbb{R} \to \mathbb{R}$ eine 2π-periodische, analytische Funktion. Nehme an, es existiert ein Gebiet $G = \mathbb{R} \times (-c, c) \subset \mathbb{C}$ mit $c > 0$, so dass f zu einer 2π-periodischen, beschränkten, holomorphen Funktion $f : G \to \mathbb{C}$ erweitert werden kann. Für den Fehler der summierten Trapezregel gilt in diesem Fall*

$$|I(f) - T_h(f)| \leq 4\pi \frac{M}{e^{cN} - 1}. \tag{5.11}$$

Hierbei bezeichnet M eine Schranke von f auf G.

Beweis. Siehe [53]. □

In diesem Fall liegt demzufolge eine exponentielle Konvergenz vor. Um eine konkrete Fehlerabschätzung der Art (5.11) für das Integral (5.9) angeben zu können, sind zunächst einige Vorbemerkungen nötig.

Zur Bestimmung einer oberen Schranke M der Funktion $\tilde{d}(t)$ auf G ist es nötig, die Norm des Terms $(r_k e^{it_j} I - A)^{-1}$ abzuschätzen. Hierbei können Pseudospektren zu Hilfe genommen werden. Eine ausführliche Darstellung zu deren Theorie und praktischer Verwendung ist in [71] gegeben. Zur Abschätzung von $\|(r_k e^{it_j} I - A)^{-1}\|_2$ kann folgende Definition genutzt werden.

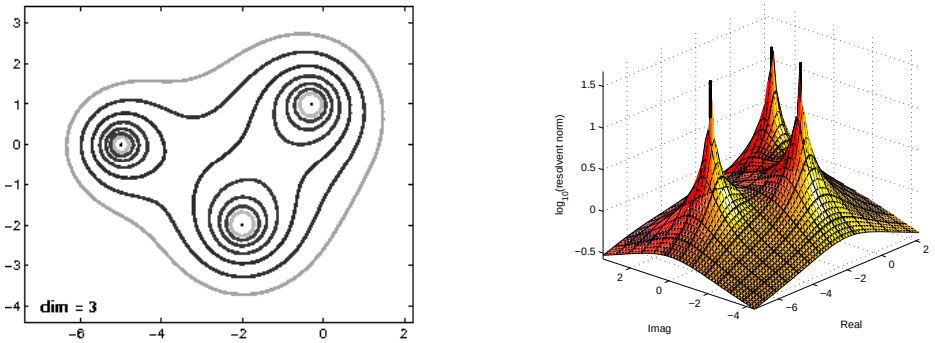
Definition 5.3.2. *Sei $A \in \mathbb{C}^{n\times n}$ und $\varepsilon > 0$ beliebig. Das ε-Pseudospektrum $\sigma_\varepsilon(A)$ von A ist die Menge aller $z \in \mathbb{C}$, so dass*

$$\|(zI - A)^{-1}\|_2 > \varepsilon^{-1} \tag{5.12}$$

gilt.

Aus (5.12) folgt insbesondere, dass $\|(zI - A)^{-1}\|_2 \leq \varepsilon^{-1}$ für alle z außerhalb des Pseudospektrums σ_ε gilt.

Beispiel 5.3.1. *Die folgenden, mit dem Paket* EigTool *[76], erzeugten Skizzen sollen der grafischen Veranschaulichung von Pseudospektren dienen.*



Die linke Abbildung zeigt von außen nach innen die σ_ε-Pseudospektren für $\varepsilon = 1, \ldots, 0.1$ der Matrix

$$A = \begin{bmatrix} -5 & 4 & 4 \\ 0 & -2-2i & 4 \\ 0 & 0 & -0.3+i \end{bmatrix}.$$

Die drei inneren Punkte $\lambda_1 = -5$, $\lambda_2 = -2-2i$ und $\lambda_3 = -0.3+i$ sind die Eigenwerte von A. Das rechte Bild zeigt die zugehörigen Werte der Norm der Resolventen $\|(zI-A)^{-1}\|_2$. Diese geht bei Annäherung von z an die Eigenwerte von A gegen unendlich.

Durch die Fortsetzung von $\tilde{d}(t)$ auf das durch (5.11) definierte Gebiet

$$G := \left\{ z = m_k + r_k e^{i(t+i\omega)} \,\middle|\, |\omega| < c, t \in [0, 2\pi] \right\},$$

ergeben sich mit $\tilde{z} = t \mp ic$ aufgrund von

$$\begin{aligned} r_k e^{i\tilde{z}} &= r_k e^{i(t\mp ic)} \\ &= r_k e^{\pm c} e^{it} \end{aligned} \tag{5.13}$$

mit $R_\pm = r_k e^{\pm c}$ die Funktionen

$$\tilde{d}_\pm(t) = f(R_\pm e^{it}) e^{\pm c} e^{it} (R_\pm e^{it} I - A)^{-1} a (1 + b^T (R_\pm e^{it} I - (A + ab^T))^{-1} a) b^T (R_\pm e^{it} I - A)^{-1}$$

auf den Rändern von G.

Bemerkung 5.3.2. *Auch auf dem erweiterten Gebiet G darf keine Singularität, d.h. weder ein Eigenwert von A oder $A + ab^T$ noch eine Singularität von f, liegen.*

Aufgrund des Maximumprinzips, siehe [55], nimmt die Funktion $\tilde{d}(t + i\omega)$ mit $|\omega| < c$ ihr Maximum genau auf einem der Ränder $\tilde{d}_\pm(t)$ an. Zur vereinfachten Schreibweise soll zunächst in folgender Fehlerbetrachtung angenommen werden, dass dieses Maximum auf dem äußeren Kreisrand mit Radius $R_+ = r_k e^c$ liegt.

Mit Hilfe von (5.12) kann damit die maximale Norm M des Integranden (5.7), eingeschränkt auf das Gebiet G, durch

$$\begin{aligned} M &= \left\| \frac{r_k}{2\pi} f(R_+ e^{it}) e^c e^{it} \tilde{B}^{-1} a (1 + b^T \tilde{B}_+^{-1} a) b^T \tilde{B}^{-1} \right\|_2 \\ &\leq \frac{r_k}{2\pi} \left[\max_{t \in [0,2\pi]} |f(R_+ e^{it})| e^c e^{it} \|\tilde{B}^{-1}\|_2 \gamma (1 + \gamma \|\tilde{B}_+^{-1}\|_2) \|\tilde{B}^{-1}\|_2 \right] \\ &\leq \frac{r_k}{2\pi} \left[\max_{t \in [0,2\pi]} |f(R_+ e^{it})| e^c e^{it} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \right] \end{aligned} \tag{5.14}$$

mit $\tilde{B} = (R_+ e^{it} I - A)$, $\tilde{B}_+ = (R_+ e^{it} I - (A + ab^T))$ und $\|\tilde{B}^{-1}\|_2 \leq \varepsilon_1^{-1}$, $\|\tilde{B}_+^{-1}\|_2 \leq \varepsilon_2^{-1}$ sowie $\gamma = \|a\|_2 \|b^T\|_2$ abgeschätzt werden. Hierbei wird vorausgesetzt, dass der Integrationskreis Γ außerhalb der zur Abschätzung verwendeten ε-Pseudospektren liegt.

Wird (5.14) in (5.11) eingesetzt, ergibt sich der folgende Fehler bei der Berechnung des Integrals (5.9) mit Hilfe der summierten Trapezregel:

$$\begin{aligned} \|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|_2 &\leq 2\pi r_k \max_{t \in [0,2\pi]} |f(R_+ e^{it})| e^c e^{it} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^{cN} - 1)} \\ &\leq 2\pi r_k \max_{t \in [0,2\pi]} |f(R_+ e^{it})| \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^N - e^{-c})}. \end{aligned} \tag{5.15}$$

Für den Fall, dass $\tilde{d}(t + i\omega)$ sein Maximum auf dem inneren Kreis $\tilde{d}_-(t)$ mit dem Radius $R_- = r_k e^{-c}$ annimmt, folgt aus der vorherigen Betrachtung die Fehlerabschätzung:

$$\begin{aligned} \|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|_2 &\leq 2\pi r_k \max_{t \in [0,2\pi]} |f(R_- e^{it})| e^{-c} e^{it} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^{cN} - 1)} \\ &\leq 2\pi r_k \max_{t \in [0,2\pi]} |f(R_- e^{it})| \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^{c(N+1)} - e^c)}. \end{aligned} \tag{5.16}$$

Hierbei ist $\|\tilde{B}^{-1}\|_2 \leq \varepsilon_1^{-1}$, $\|\tilde{B}_+^{-1}\|_2 \leq \varepsilon_2^{-1}$ mit $\tilde{B} = (R_- e^{it} I - A)$, $\tilde{B}_+ = (R_- e^{it} I - (A + ab^T))$.

Aus diesen Abschätzungen folgt, dass die Anwendung der summierten Trapezregel auf das Integral (5.9) zu einer exponentiell abfallenden Fehlerentwicklung führt. Mit dem Satz von Eckart-Young-Schmidt-Mirsky, siehe [3], ergibt sich weiterhin ein ebensolcher exponentieller Abfall für die Singulärwerte von $I(\tilde{d}(t))$.

Bemerkung 5.3.3. *Da auch bei der Anwendung von Quadraturverfahren höherer Ordnung, wie beispielsweise der summierten Simpsonregel, keine signifikant bessere Fehlerentwicklung erreicht werden kann, siehe Kapitel 5.6, beschränkt sich diese Arbeit auf die Verwendung der summierten Trapezregel.*

5.3.2 Effiziente Berechnung

Bestimmung des Integrationswegs

Wie bereits beschrieben, muss aufgrund der Definition des Cauchy-Integrals der Integrationskreis Γ sowohl die Spektren $\sigma(A)$ und $\sigma(A+ab^T)$ umschließen. Aus (5.15) folgt, dass der Approximationsfehler hauptsächlich vom Abstand dieses Kreises zu den Spektren sowie von der Funktion f abhängt. Zum einen darf der Radius des Integrationskreises nicht zu klein gewählt werden, da er in diesem Fall zu dicht an $\sigma(A)$ bzw. $\sigma(A + ab^T)$ liegt und somit sehr kleine Werte ε_1 bzw. ε_2 auftreten. Zum anderen kann auch ein zu großer Radius aufgrund des Terms $r_k \max(f(r_k e^{c+it}))$ im Zähler von (5.15) zu einer schlechten Approximation führen. Dies tritt beispielsweise im Fall der Exponentialfunktion auf, in welchem $f(r_k e^{c+it}) = e^{r_k e^{c+it}}$ ist, was für einen großen Radius r_k sehr schnell wächst.

Aus diesen Betrachtungen ergibt sich die Fragestellung, wie die optimalen Werte für Mittelpunkt und Radius des Integrationskreises konkret bestimmt werden können. Da eine explizite Berechnung der Spektren bzw. auch der ε-Pseudospektren von A und $A+ab^T$ numerisch zu aufwendig ist, sind diese für die praktische Verwendung ungeeignet. Stattdessen kann auf den Wertebereich der Matrizen A und $A + ab^T$ zurückgegriffen werden, welcher wie folgt definiert ist:

Definition 5.3.3. *Sei $A \in \mathbb{C}^{n\times n}$, so ist durch*

$$W(A) = \left\{ \frac{x^* A x}{x^* x} \middle| x \in \mathbb{C}^n, x \neq 0 \right\}$$

der Wertebereich von A definiert.

Da für das Spektrum $\sigma(A) \subseteq W(A)$ gilt, bietet sich der Schwerpunkt der äußeren Hülle von $W(A)$ und $W(A + ab^T)$ als eine geeignete Wahl für den Mittelpunkt des Integrationskreises an. Des Weiteren lässt sich daraus ableiten, dass der Radius des Kreises so bestimmt werden sollte, dass dieser die beiden Wertebereiche umschließt. Damit wird gewährleistet, dass die Spektren $\sigma(A)$ und $\sigma(A + ab^T)$ im Inneren der Integrationskurve liegen.

Die numerische Berechnung der Wertebereiche kann, wie in [16, 51] dargestellt wird, erfolgen. Diese Algorithmen basieren auf folgender Idee.

Nach der Unterteilung des Winkels 2π durch M Diskretisierungspunkte wird zu jedem Winkel $\alpha_j = j \cdot \frac{2\pi}{M}$ für $j = 0, \ldots, M$ zunächst die gedrehte Matrix

$$A_{\alpha_j} = e^{-i\alpha_j} A \tag{5.17}$$

berechnet. Im nächsten Schritt werden von jeder der hermitischen Matrizen

$$H_j = \frac{A_{\alpha_j} + A_{\alpha_j}^H}{2}$$

die Eigenwerte $\lambda(H_j)$ bestimmt. Durch den größten Eigenwert kann der Wertebereich $W(A)$ in Richtung des Winkels α_j angenähert werden. Eine Approximation des gesamten Wertebereichs $W(A)$ in der komplexen Ebene ergibt sich durch das Bilden der konvexen Hülle aller so berechneten Punkte.

Dieses Vorgehen führt mit einer ausreichend großen Anzahl von k Stützstellen zu einer sehr guten Approximation des Wertebereichs $W(A)$. Der Nachteil dieses Algorithmus ist die hohe numerische Komplexität $\mathcal{O}(n^3)$, welche durch die Berechnung der Eigenwerte von H_j zu jedem Winkel α_j hervorgerufen wird. Je nachdem mit welcher Genauigkeit $W(A)$ und $W(A + ab^T)$ angenähert werden sollen, kann der zur Berechnung benötigte numerische Aufwand jedoch wie folgt reduziert werden:

- Da zu jedem Winkel α_j nur der größte Eigenwert der Matrix H_j gesucht ist, kann dieser zum Beispiel durch Gerschgorin-Kreise, siehe Abschnitt 2.9, approximiert werden. Dies reduziert den benötigten Rechenaufwand auf $\mathcal{O}(n^2)$.

- Anstelle der Verwendung von k Drehungen kann der Wertebereich durch ein diesen umschließendes Rechteck approximiert werden. Die Seiten des Rechtecks sind hierbei durch die größten Eigenwerte von $\frac{A+A^H}{2}$, $\frac{A-A^H}{2}$, $\frac{-A+A^H}{2}$ und $\frac{-A-A^H}{2}$ definiert. Diese Eigenwerte können erneut durch Gerschgorin-Kreise bzw. mit wenigen Lanczos-Schritten, siehe [29], bestimmt werden.

Lösung der geshifteten Gleichungssysteme

Neben der nichttrivialen Wahl des Integrationswegs Γ verbleibt als hauptsächlicher Berechnungsaufwand das Lösen der Gleichungssysteme $(z_j I - A)^{-1} a$ und $b^T (z_j I - A)^{-1}$ mit $j = 0, \ldots, N$ bei der Funktionsauswertung zu jedem Quadraturpunkt der summierten Trapezregel. Hierfür bietet sich die Verwendung des Restarted-GMRES Verfahrens an, siehe [57]. In [21] wird gezeigt, dass dabei für jedes weitere Gleichungssystem $(z_j I - A)^{-1} a$ im Wesentlichen nur eine Matrix-Vektor Multiplikation zusätzlich zur Lösung eines der Systeme $(z_0 I - A)^{-1} a$ nötig ist. Dies beruht auf der Gleichheit der Krylovräume der Matrizen $(z_j I - A)$. Der Nachteil dieses Vorgehens ist, dass es nicht möglich ist, Vorkonditionierer zu verwenden und somit im allgemeinen Fall ein schlechtes Konvergenzverhalten zu erwarten ist. Im ungünstigsten Fall ist es nötig, den kompletten Krylovraum zu berechnen, was einen Rechenaufwand von einmalig $4n^3 + \mathcal{O}(n^2)$ Operationen erfordert. Dieses Vorgehen kann analog mit den Gleichungssystemen $b^T (z_j I - A)^{-1}$ durchgeführt werden.

Eine weitere Möglichkeit ist, zunächst eine Hessenbergreduktion der Matrix

$$Q^T (\underbrace{z_0 I - A}_{\tilde{A}}) Q = H \tag{5.18}$$

vorzunehmen. Hierbei ist H eine obere Hessenbergmatrix. Des Weiteren ist Q eine Orthogonalmatrix, welche durch wiederholte Householder-Spiegelungen, siehe Kapitel 2.1.2, erzeugt werden kann. Zur Lösung des Gleichungssystems $QHQ^T x_j = a$ ist allerdings eine explizite Darstellung von Q nicht erforderlich. Damit sind zunächst zur Berechnung der Hessenbergform (5.18) $\frac{10}{3} n^3$ Additionen und Multiplikationen nötig, siehe [29]. Die Berechnung von x_0 ist daraufhin mit einem Aufwand von $\mathcal{O}(n^2)$ möglich. Der Vorteil in diesem Vorgehen besteht im Weiteren darin, dass die Hessenbergzerlegungen der geshifteten Systeme $(\tilde{z}_j I + \tilde{A})$ sofort aus (5.18) als

$$Q^T (\tilde{z}_j I + \tilde{A}) Q = H + (\tilde{z}_j - z_0) I$$

abgeleitet werden können. Somit kann jeder weitere Lösungsvektor x_j mit $\mathcal{O}(n^2)$ Operationen berechnet werden. Des Weiteren ist auch die Lösung der Gleichungssysteme $b^T (z_j I - A)^{-1}$ mit Hilfe der Zerlegung (5.18) in einer Komplexität von jeweils $\mathcal{O}(n^2)$ möglich.

Jede dieser Methoden erfordert maximal einen kubischen Rechenaufwand zur Lösung eines der Gleichungssysteme. Sämtliche weiteren geshifteten Systeme sind daraufhin in $\mathcal{O}(n^2)$ lösbar. Dies ist schon bei relativ geringer Stützstellenanzahl signifikant weniger, als die separate Lösung der einzelnen Systeme in jeweils $\mathcal{O}(n^3)$ Operationen.

Bemerkung 5.3.4. *In diesem Zusammenhang wird deutlich, dass für die praktische Berechnung die Darstellung (5.6) des Integranden $d(z)$ zu bevorzugen ist. Im Gegensatz dazu würde die Verwendung von (5.7) die Lösung einer weiteren Folge von Gleichungssystemen mit den Systemmatrizen $(z_j I - A - ab^T)$ erfordern.*

Wahl der Quadraturpunkte

Eine weitere Fragestellung ist, wieviele Quadraturpunkte verwendet werden müssen, um eine gewünschte Genauigkeit zu erhalten. Da eine explizite Berechnung der Fehlerschranken (5.15) und (5.16) extrem aufwendig ist, sind diese für die praktische Anwendung ungeeignet. Anstelle dessen kann wie folgt ein Fehlerschätzer konstruiert werden. Mit $\|\cdot\|$ sei in diesem Abschnitt eine beliebige, in der praktischen Anwendung effizient berechenbare Norm bezeichnet.

Aufgrund der vorangegangenen Betrachtungen ist bekannt, dass der Integrationsfehler $E(N) = \|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|$ in Abhängigkeit von der Stützstellenzahl exponentiell fällt. Es folgt aus (5.11), dass

$$E(N) = c_1(e^{c_2 N} - 1)^{-1} \tag{5.19}$$

ist.

Zunächst werden zu verschiedener Anzahl an Quadraturpunkten N_i die Trapezsummen T_{h_i} mit $h_i = \frac{b-a}{N_i}$ berechnet. Eine mögliche Wahl ist hierbei zum Beispiel $N_i = 2^i$, für $i \in \mathbb{N}_+$. Der Vorteil der Nutzung von Zweierpotenzen liegt darin, dass die für die Trapezsumme T_{h_i} berechneten Summanden bei Verfeinerung der Diskretisierung zur Berechnung von $T_{h_{i+1}}$ wiederverwendet werden können.

Anhand der so berechneten Werte (N_i, T_{h_i}) können durch die Lösung des nichtlinearen Ausgleichsproblems

$$\|I - T_{h_i}\| \approx c_1(e^{c_2 N_i} - 1)^{-1} \tag{5.20}$$

mit $n_1 \leq i < n_2$, $1 \leq n_1$ und $n_2 > n_1 + 1$ die Konstanten c_1 und c_2 geschätzt werden. Hierzu kann als Approximation des exakten Wertes I, der mit der bis dahin größten Stützstellenzahl N_{n_2} berechnete Wert verwendet werden.

Sind die Konstanten c_1 und c_2 bekannt, kann aus (5.19) die benötigte Anzahl an Quadraturpunkten bestimmt werden, welche benötigt wird um die gewünschte Genauigkeit zu erreichen.

Anstelle von (5.20) sollten zur Lösung des nichtlinearen Ausgleichsproblems die Gleichungen

$$\ln \|I - T_{h_i}\| \approx \ln(c_1(e^{c_2 N_i} - 1)^{-1}) \tag{5.21}$$

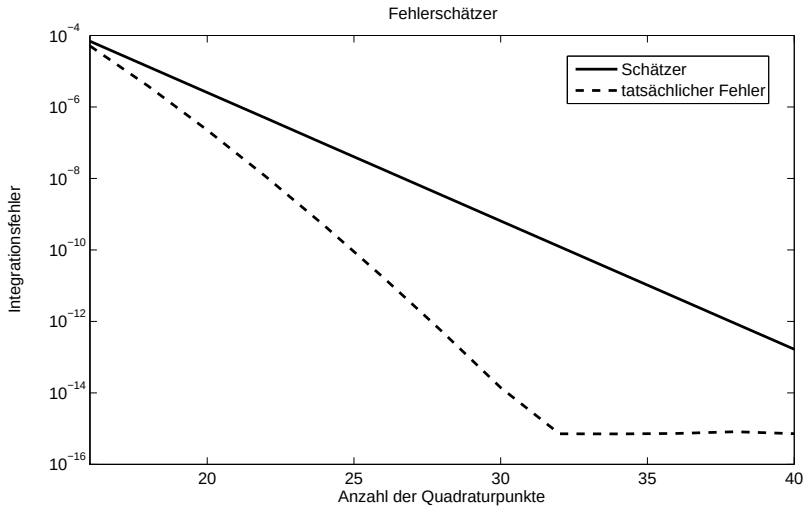


Abbildung 5.1: Schätzung des Integrationsfehlers

verwendet werden. In Beispiel 5.3.2 wird dieser Fehlerschätzer praktisch angewendet.

Beispiel 5.3.2. *Für $N = 4, 8, 16$ wurden die zugehörigen Trapezsummen $T_{2\pi/4} = 0.78509$, $T_{2\pi/8} = 0.12862$ und $T_{2\pi/16} = 0.000051$ berechnet. Das hieraus abgeleitete nichtlineare Ausgleichsproblem wurde mit der* MATLAB *Routine* lsqnonlin *gelöst. Hierbei ergaben sich $c_1 = 38.339$ und $c_2 = 0.82691$. Die daraus resultierende Fehlerschätzung sowie der tatsächlich berechnete Fehler sind in Abbildung 5.1 dargestellt. Wie in den meisten durchgeführten Tests, ist auch in diesem Beispiel zu erkennen, dass der tatsächlich erhaltene Fehler meist deutlich besser als der geschätzte Wert ist.*

Bemerkung 5.3.5. *Durch die Vernachlässigung der -1 kann (5.21) zum linearen Ausgleichsproblem $\ln \|I - T_{h_i}\| \approx \ln c_1 - c_2 N_i$ vereinfacht werden.*

Vereinfachungen im reellen Fall

Im Fall eines reellen Aufdatierungsproblems, d.h. $A \in \mathbb{R}^{n\times n}$ und $a, b \in \mathbb{R}^n$, kann die Integration vereinfacht werden. Aufgrund der Symmetrie des Spektrums ist es hierbei ausreichend, zunächst über den oberen Halbkreis $\tilde{\Gamma} : [0, \pi] \to \mathbb{C}, t \to m_k + r_k e^{it}$ zu integrieren. Die hierbei erhaltene Trapezsumme sei mit $\tilde{T}_h(\tilde{d}(t))$ bezeichnet. Daraus lässt sich die Approximation des Integrals über den gesamten Kreis Γ durch die Verdopplung des Realteils von $\tilde{T}_h(\tilde{d}(t))$, d.h.

$$T_h(\tilde{d}(t)) = 2\,\mathrm{Real}(\tilde{T}_h(\tilde{d}(t)))$$

berechnen. Aufgrund dieser Vereinfachung halbiert sich die Anzahl der zu lösenden Gleichungssysteme. Weiterhin folgt daraus, dass der numerische Rang von K in diesem Fall maximal $N/2$ sein kann.

5.3.3 Anwendung auf die Matrixexponentialfunktion

Die in den vorherigen Abschnitten beschriebenen Resultate lassen sich intuitiv auf die Exponentialfunktion übertragen. Aus (5.9) folgt in diesem Fall

$$e^{A+ab^T} - e^A = \frac{r_k}{2\pi} \int_0^{2\pi} e^{r_k e^{it}+it} (r_k e^{it} I - A)^{-1} a (1 + b^T (r_k e^{it} I - (A + ab^T))^{-1} a) b^T (r_k e^{it} I - A)^{-1} \, \mathrm{dt}, \tag{5.22}$$

sowie die zur Lösung des Integrals zu berechnende Trapezsumme

$$T_h(\tilde{d}(t)) = \frac{r_k}{2\pi} h \sum_{j=0}^{N-1} f(r_k e^{it_j}) e^{it_j} (1 + b^T (r_k e^{it_j} I - (A + ab^T))^{-1} a)(r_k e^{it_j} I - A)^{-1} a b^T (r_k e^{it_j} I - A)^{-1}.$$

Aus (5.11) ergibt sich zum einen die Fehlerabschätzung der summierten Trapezregel als

$$\|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|_2 \leq 2\pi r_k \max_{t \in [0,2\pi]} e^{R_+ e^{it} + c + it} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^{cN} - 1)},$$

für den äußeren Radius R_+, welche ihr Maximum

$$\begin{aligned} \|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|_2 &\leq 2\pi r_k e^{R_+ + c} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \frac{1}{(e^{cN} - 1)} \\ &\leq \frac{2\pi r_k e^{R_+}}{(e^N - e^{-c})} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right) \end{aligned} \tag{5.23}$$

bei $t = 0$ annimmt.
Zum anderen gilt für den inneren Radius R_- sowie $t = 0$:

$$\|I(\tilde{d}(t)) - T_h(\tilde{d}(t))\|_2 \leq \frac{2\pi r_k e^{R_-}}{(e^{c(N+1)} - e^c)} \left(\frac{\gamma}{\varepsilon_1^2} + \frac{\gamma^2}{\varepsilon_1^2 \varepsilon_2} \right). \tag{5.24}$$

Dies zeigt, dass im Fall der Matrixexponentialfunktion die Verwendung der summierten Trapezregel eine exponentiell abfallende Fehlerentwicklung liefert. Aus (5.23) ist weiter zu erkennen, dass mit größerem Radius $R_+ = r_k e^c$ die Anzahl der Stützstellen N erhöht werden muss, um eine gute Approximation zu erhalten. Wird hingegen r_k zu klein gewählt und folgt daraus, dass $c \ll 1$ ist, muss auch in diesem Fall eine größere Stützstellenanzahl verwendet werden.

Wie Mittelpunkt und Radius des Integrationskreises für verschiedene Beispiele gewählt werden sollten bzw. welche Auswirkungen deren Variation hat, wird im Kapitel 5.6 anhand numerischer Resultate dargestellt.

5.4 Der Scaling und Squaring Algorithmus

Die Scaling und Squaring Methode ist der meistgenutzte Algorithmus zur Berechnung der Matrixexponentialfunktion. Die Grundidee dieses Verfahrens, vgl. [47], wird im Folgenden zunächst vorgestellt. Weiterhin werden zwei der zahlreichen Modifikationen der ursprünglichen Methode, siehe [2, 46], dargestellt.

Das Scaling und Squaring Verfahren beruht auf der Eigenschaft

$$e^A = (e^{\frac{A}{\sigma}})^\sigma, \text{ für } A \in \mathbb{C}^{n\times n}, \sigma \in \mathbb{C} \tag{5.25}$$

sowie der Tatsache, dass e^A sehr gut durch die Padé-Approximation, vgl. Kapitel 2.8, nahe des Ursprungs, d.h. für kleine $\|A\|_1$, angenähert werden kann. Die Grundidee ist, ein $\sigma = 2^s$ so zu wählen, dass die Norm von $\tilde{A} = \frac{1}{\sigma}A$ von der Ordnung 1 ist. Danach kann $e^{\tilde{A}}$ durch die Padé-Approximation

$$e^{\frac{1}{\sigma}A} \approx r_{km}(\frac{1}{\sigma}A) \tag{5.26}$$

effizient approximiert werden. Abschließend erhält man durch wiederholtes Quadrieren die Lösung

$$e^A \approx r_{km}(\tilde{A})^s.$$

Die Padé-Approximation $e^A \approx r_{km}(A) = p_{km}(A) \cdot q_{km}(A)^{-1}$ der Exponentialfunktion ist für alle k und m explizit durch

$$\begin{aligned} p_{km}(A) &= \sum_{j=0}^{k} \frac{(k+m-j)!k!}{(k+m)!(k-j)!}\frac{A^j}{j!} \quad \text{und} \\ q_{km}(A) &= \sum_{j=0}^{m} \frac{(k+m-j)!m!}{(k+m)!(m-j)!}\frac{(-A)^j}{j!} \end{aligned} \tag{5.27}$$

gegeben, vgl. [47].

Lemma 5.4.1. *Für den skalaren Fall gilt die Fehlerabschätzung*

$$e^x - r_{km} = (-1)^m \frac{k!m!}{(k+m)!(k+m+1)!}x^{k+m+1} + \mathcal{O}(x^{k+m+1}).$$

Des Weiteren ist der exakte Fehler für matrixwertige Argumente $A \in \mathbb{C}^{n\times n}$ durch

$$e^A - r_{km}(A) = \frac{(-1)^m}{(k+m)!}A^{k+m+1}q_{km}(A)^{-1}\int_0^1 e^{tA}(1-t)^k t^m dt.$$

gegeben.

Beweis. Siehe [63]. □

Üblicherweise wird der Polynomgrad von p_{km} und q_{km} gleich gewählt. Dies ist darin begründet, dass eine Approximation mit r_{km}, wobei $k \neq m$ ist, ungenauer ist als die Approximation mit $r_j = r_{jj}$ und $j = \max(k, m)$. Demzufolge soll im Weiteren zur Approximation der Exponentialfunktion die rationale Funktion $r_m(A) = p_m(A) \cdot q_m(A)^{-1}$ verwendet werden.

Hieraus folgt die Problemstellung, wie eine geeignete Wahl von s und m aussieht. Auf der einen Seite ist eine geringe Norm von $\tilde{A}$, was einem großen s entspricht, sowie ein hoher Polynomgrad m nötig, um eine möglichst genaue Approximation $e^A \approx r_m^{2^s}$ zu erhalten. Im Gegensatz dazu führen sowohl große s wie auch m zu erheblich höherem Rechenaufwand, speziell im Fall der matrixwertigen Exponentialfunktion.

Fehlerbetrachtung

Folgt man der Fehleranalyse in [47], so ergeben sich zu verschiedenen Polynomgraden m die in Tabelle 5.1 dargestellten Normen $\Theta_m = \|\tilde{A}\|_1$, welche $\tilde{A}$ maximal haben darf, um die Genauigkeit der Lösung in Größenordung der Machinengenauigkeit $\epsilon = 2^{-53}$ zu garantieren. Aus der ursprünglichen Matrix A, dem Polynomgrad m und der maximal zugelassenen Matrixnorm Θ_m ergibt sich damit der Skalierungsparameter

$$s = \begin{cases} \left\lceil \frac{\log_2 \|A\|_1}{\Theta_m} \right\rceil, & \text{wenn } \|A\|_1 > \Theta_m \\ 0, & \text{sonst} \end{cases},$$

welcher minimal nötig ist um eine Approximationsgenauigkeit im Rahmen der Maschinengenauigkeit ϵ zu gewährleisten.

m	3	5	7	9	11	13	15	17	19
Θ_m	$1.5e-2$	$2.5e-1$	$9.5e-1$	$2.1e0$	$3.6e0$	$5.4e0$	$7.3e0$	$9.4e0$	$1.2e1$

Tabelle 5.1: Maximal zulässige Norm von A zum Polynomgrad m

Berechnungsaufwand

Um den Aufwand zur Berechnung der Matrixexponentialfunktion, basierend auf [47], anzugeben, wird zunächst eine effiziente Darstellung der Polynome p_m und q_m vorgestellt.

Das Polynom $p_m = \sum_{i=0}^{m} b_i x^i$ ist wie folgt explizit gegeben:

- Für geraden Polynomgrad gilt:

$$\begin{aligned} p_{2m} &= \underbrace{b_{2m}A^{2m} + \cdots + b_2A^2 + b_0I}_{U_{2m}} + \underbrace{A(b_{2m-1}A^{2m-2} + \cdots + b_3A^2 + b_1I)}_{V_{2m-1}} \\ &= U_{2m} + V_{2m-1}. \end{aligned} \tag{5.28}$$

- Für ungeraden Polynomgrad gilt:

$$\begin{aligned} p_{2m+1} &= \underbrace{A(b_{2m+1}A^{2m} + \cdots + b_3A^2 + b_1I)}_{U_{2m+1}} + \underbrace{b_{2m}A^{2m} + \cdots + b_2A^2 + b_0I}_{V_{2m}} \\ &= U_{2m+1} + V_{2m}. \end{aligned} \tag{5.29}$$

Aufgrund von (5.27) folgt $q_m(A) = p_m(-A)$ und somit sowohl für gerade und ungerade m, dass

$$q_{2m} = U_{2m} - V_{2m-1} \quad \text{bzw.} \quad q_{2m+1} = -U_{2m+1} + V_{2m}$$

ist. Die Koeffizienten b_i sind durch (5.27) eindeutig definiert.

Bemerkung 5.4.1. *Im Folgenden wird von einem ungeraden Polynomgrad ausgegangen, um zur besseren Lesbarkeit auf Fallunterscheidungen verzichten zu können. In diesem Zusammenhang soll ebenfalls auf die Verwendung der unteren Indizes von U und V verzichtet werden.*

Zur Bestimmung von r_m ist letztendlich die Matrixgleichung

$$q_m r_m = p_m \quad \Longleftrightarrow \quad (-U + V)r_m = U + V \tag{5.30}$$

zu lösen. Somit besteht der hauptsächliche Aufwand des Scaling und Squaring Algorithmus aus

- der Potenzierung der Matrix A zur Berechnung von p_m, wobei die Anzahl der dazu nötigen Matrixprodukte im Folgenden mit π_m bezeichnet wird,
- der Lösung der Matrixgleichung (5.30) zur Berechnung von r_m,
- und dem s-fachen Quadrieren von r_m zur Rückskalierung.

Die Anzahl benötigter Matrixmultiplikationen in Abhängigkeit von m und s sind folglich insgesamt

$$\pi_m + s = \pi_m + \max(\lceil \log_2 \|A\|_1 - \log_2 \Theta_m \rceil, 0). \tag{5.31}$$

Das Ziel ist eine Aussage zu treffen, welcher Polynomgrad m in Bezug auf den Berechnungsaufwand sowie die Approximationsgenauigkeit am geeignetsten ist. Dazu können in (5.31) die Multiplikationen zur Lösung der Matrixgleichung (5.30) außen vor gelassen werden, da deren Anzahl unabhängig von m und s konstant ist. Für eine bessere Vergleichbarkeit kann weiterhin sowohl das Maximum als auch der konstante Term $\|A\|_1$ in (5.31) vernachlässigt werden. Damit ergibt sich das folgende, nur von m abhängige Maß M_m für den Gesamtaufwand des Algorithmus:

$$M_m = \pi_m - \log_2 \Theta_m.$$

Die Werte π_m sowie M_m sind in Tabelle 5.2 für einige ausgewählte m dargestellt. Die

m	3	5	7	9	11	13	15	17	19
π_m	2	3	4	5	6	6	7	7	8
M_m	8.1	5.0	4.1	3.9	4.2	3.6	4.1	3.8	4.5

Tabelle 5.2: Aufwand in Abhängigkeit vom Polynomgrad

teilweise Diskrepanz zwischen π_m und zugehörigem M_m ergibt sich aus den benötigten Normen Θ_m, vgl. Tabelle 5.1, und der damit verbundenen Anzahl an Skalierungsschritten. Der optimale Polynomgrad zur Berechnung der Matrixexponentialfunktion mittels des Scaling und Squaring Algorithmus bei einer Genauigkeit von $\epsilon = 2^{-53}$ beträgt somit $m = 13$.

Bemerkung 5.4.2. *In Tabelle 5.2 wird von einer möglichst effizienten Durchführung der Potenzierung der Matrix A in p_m ausgegangen. Für den Fall $m = 13$ sieht diese wie folgt aus:*

$$\begin{aligned} U &= A\left[A^6\left(b_{13}A^6 + b_{11}A^4 + b_9A^2\right) + b_7A^6 + b_5A^4 + b_3A^2 + b_1I\right] \\ V &= A^6\left(b_{12}A^6 + b_{10}A^4 + b_8A^2\right) + b_6A^6 + b_4A^4 + b_2A^2 + b_0I. \end{aligned} \tag{5.32}$$

Hierbei sind die Matrixprodukte $A^2 = AA$, $A^4 = A^2A^2$, $A^6 = A^4A^2$ sowie die drei Produkte von A^6 bzw. A mit den zugehörigen Klammerausdrücken zu bilden.

Scaling und Squaring Implementierung

Der Scaling und Squaring Algorithmus zur Berechnung von $E = e^A$ mit $A \in \mathbb{R}^{n\times n}$ ist durch folgenden, aus [47] entnommenen Pseudo-Code gegeben:

1: **for** $m = [3, 5, 7, 9]$ **do**
2: **if** $\|A\|_1 \leq \Theta_m$ **then**
3: Berechnung von U und V durch (5.28) und (5.29).
4: Lösung von $(-U + V)E = U + V$.
5: QUIT
6: **end if**
7: **end for**
8: $A \leftarrow \frac{A}{2^s}$ mit dem minimalen Integer $s \geq 0$, so dass $\|\frac{A}{2^s}\|_1 \leq \Theta_{13}$.
9: Berechnung von $A_2 = A^2$, $A_4 = A_2^2$ und $A_6 = A_2A_4$.
10: Berechnung von U und V durch (5.32).
11: Lösung von $(-U + V)r_{13} = U + V$.
12: Berechnung von $E = r_{13}^{2^s}$ durch wiederholtes Quadrieren.

Erweiterungen des Scaling und Squaring Algorithmus

An dieser Stelle sollen zwei Modifikationen des Scaling und Squaring Algorithmus kurz vorgestellt werden.

In [46] werden zur Reduktion der Anzahl der Skalierungen folgende Modifikationen vorgeschlagen:

(i) Verringerung der Norm $\|A\|_1$ durch (1) $A \leftarrow A - \mu I$ und (2) $A \leftarrow D^{-1}AD$. Dabei sind μ und D so zu wählen, dass $\|A\|_1$ reduziert wird und D^{-1} sowie Matrixprodukte mit D bzw. D^{-1} effizient berechnet werden können.

(ii) Durchführung des Scaling und Squaring Algorithmus zur Berechnung von $X = r_m^{2^s}$ wie zuvor beschrieben.

(iii) Durchführung der zu (i) gehörigen Rücktransformation bezüglich X, d.h.

$$e^A = e^\mu DXD^{-1}.$$

In [2] wird eine Vorgehensweise dargestellt, in welcher der tatsächlich verwendete Polynomgrad m sowie der Skalierungsparameter s erst während der Potenzierung der

Matrix A bestimmt wird. Dies geschieht anhand von Normschätzungen und beruht auf Schranken $\tilde{\Theta}_m$, welche auf $\|A^k\|_1^{\frac{1}{k}}$ anstelle von $\|A\|_1$ basieren. Diese Modifikation führt bei zahlreichen in [2] vorgestellten Beispielen sowohl zu einer besseren Lösungsgenauigkeit als auch zu geringerem Berechnungsaufwand im Vergleich zum ursprünglichen Algorithmus.

5.5 Aufdatierung mittels Scaling und Squaring

In diesem Kapitel wird eine neu entwickelte Vorgehensweise beschrieben, wie der Scaling und Squaring Algorithmus auf das Rang-1 Aufdatierungsproblem (5.1)–(5.3) angewendet werden kann. Ziel ist demnach die Berechnung von $E_+ = e^{A+ab^T} \in \mathbb{R}^{n\times n}$ unter der Annahme, dass $E = e^A$ bereits bekannt ist. Unter gewissen Voraussetzungen ist diese Aufdatierung der Matrixexponentialfunktion mit einem Aufwand von $\mathcal{O}(n^2)$ Operationen möglich. Zur Vereinfachung der Notation sei im Folgenden angenommen, dass $\tilde{A} = A$ ist.

Analog zu (5.26) kann E_+ mit Hilfe der Padé-Approximation

$$e^{A+ab^T} \approx r_m^+(A+ab^T) = p_m^+(A+ab^T)\cdot q_m^+(A+ab^T)^{-1}$$

approximiert werden. Ähnlich zum vorherigen Kapitel werden für p_m^+ und q_m^+ die Polynome

$$\begin{aligned} p_m^+ &= \sum_{i=0}^{m} b_i (A+ab^T)^i \\ &= U_+ + V_+ \end{aligned}$$

sowie

$$q_m^+ = -U_+ + V_+$$

verwendet.

Für den aufdatierten Fall folgt daraus für p_m^+, dass

$$\begin{aligned} p_m^+ &= \sum_{i=0}^{m} b_i (A+ab^T)^i \\ &= \underbrace{\sum_{i=0}^{m} b_i A^i}_{p_m} + \underbrace{\sum_{i=0}^{m} b_i \overbrace{\left[(A+ab^T)^i - A^i\right]}^{L_i}}_{p_k} \end{aligned} \tag{5.33}$$

ist. Hierbei ist durch p_k die Differenz zwischen den alten sowie aufdatierten Zählern der Padé-Approximationen $e^A \approx r_m$ bzw. $e^{A+ab^T} \approx r_m^+$ bezeichnet.

Für ungeraden Polynomgrad folgt damit

$$\begin{aligned} p_{2m+1} &= \underbrace{(A+ab^T)(b_{2m+1}(A+ab^T)^{2m}+\cdots+b_3(A+ab^T)^2+b_1I)}_{U_+}+ \\ &\quad +\underbrace{b_{2m}(A+ab^T)^{2m}+\cdots+b_2(A+ab^T)^2+b_0I}_{V_+}. \end{aligned}$$

Hieraus lassen sich die folgenden Änderungen der Matrizen U und V zu den aufdatierten Werten U_+ sowie V_+ ableiten:

$$\begin{aligned} U_+ &= (A+ab^T)(b_{2m+1}(A+ab^T)^{2m}+\cdots+b_3(A+ab^T)^2+b_1I) \\ &= A(b_{2m+1}(A+ab^T)^{2m}+\cdots+b_3(A+ab^T)^2+b_1I)+ \\ &\quad +ab^T(b_{2m+1}(A+ab^T)^{2m}+\cdots+b_3(A+ab^T)^2+b_1I) \\ &= \underbrace{A(b_{2m+1}A^{2m}+\cdots+b_3A^2+b_1I)}_{U}+ \\ &\quad +\underbrace{A(b_{2m+1}L_{2m}+\cdots+b_3L_2)}_{U_{k_1}}+ \\ &\quad +\underbrace{ab^T(b_{2m+1}(A+ab^T)^{2m}+\cdots+b_3(A+ab^T)^2+b_1I)}_{U_{k_2}} \\ &= U+U_{k_1}+U_{k_2}, \end{aligned} \tag{5.34}$$

sowie

$$\begin{aligned} V_+ &= b_{2m}(A+ab^T)^{2m}+\cdots+b_2(A+ab^T)^2+b_0I \\ &= \underbrace{b_{2m}A^{2m}+\cdots+b_2A^2+b_0I}_{V}+\underbrace{b_{2m}L_{2m}+\cdots+b_2L_2}_{V_k} \\ &= V+V_k, \end{aligned} \tag{5.35}$$

wobei die Matrizen $L_i \in \mathbb{R}^{n\times n}$ wie in (5.33) definiert sind, d.h. $(A+ab^T)^i = A^i + L_i$. Insgesamt gilt damit für ungeraden Index, dass

$$p^+_{2m+1} = U_+ + V_+ = \underbrace{(U+V)}_{p_{2m+1}} + U_{k_1} + U_{k_2} + V_k$$

sowie

$$q^+_{2m+1} = -U_+ + V_+ = \underbrace{(-U+V)}_{q_{2m+1}} - U_{k_1} - U_{k_2} + V_k$$

ist. Für Approximationen mit geradem Index p^+_{2m} gilt ein analoges Resultat. Zur Veranschaulichung dieser allgemeinen Darstellung soll in folgendem Beipiel p^+_3 explizit angegeben werden.

Beispiel 5.5.1. *Im Fall $m = 3$ ist*

$$p_3^+ = (A + ab^T)(b_3(A + ab^T)^2 + b_1 I) + b_2(A + ab^T)^2 + b_0 I. \tag{5.36}$$

Dies ist nach Gleichung (5.34) und (5.35) als die Aufdatierung des Polynoms p_3

$$p_3^+ = \underbrace{A(b_3A^2 + b_1I)}_{U} + \underbrace{b_3AL_2}_{U_{k_1}} + \underbrace{ab^T(b_3(A + ab^T)^2 + b_1I)}_{U_{k_2}} + \underbrace{b_2A^2 + b_0I}_{V} + \underbrace{b_2L_2}_{V_k},$$

mit $L_2 = ab^TA + Aab^T + ab^Tab^T$ darstellbar.

Rang der Aufdatierung

Aus (5.34) lässt sich sofort ableiten, dass

$$\operatorname{rang}(U_{k_2}) = 1 \tag{5.37}$$

ist. Zur Bestimmung des Rangs von U_{k_1}, V_k sowie deren Summe bzw. Differenz ist zunächst eine allgemeine Darstellung von L_i nötig. Zur besseren Anschauung soll zunächst L_i für $i = 1, 2, 3$ explizit dargestellt werden:

$$\begin{aligned}
(A + ab^T)^1 &= A + \underbrace{ab^T}_{L_1} \\
(A + ab^T)^2 &= A^2 + \underbrace{ab^TA + Aab^T + (ab^T)^2}_{L_2} \\
(A + ab^T)^3 &= A^3 + \underbrace{Aab^TA + ab^TA^2 + (ab^T)^2A + A(ab^T)^2 + ab^TAab^T + (ab^T)^3}_{L_3}.
\end{aligned}$$

Allgemein gilt, dass

$$(A + ab^T)^i = A^i + \underbrace{\underbrace{\begin{bmatrix} a & Aa & A^2a & \cdots & A^{i-1}a \end{bmatrix}}_{\in\mathbb{R}^{n\times i}} \underbrace{\begin{bmatrix} c_{1,1} & \cdots & c_{1,i} \\ \vdots & & \vdots \\ c_{i,1} & \cdots & c_{i,i} \end{bmatrix}}_{=C\in\mathbb{R}^{i\times i}} \underbrace{\begin{bmatrix} b^T \\ b^TA \\ b^TA^2 \\ \vdots \\ b^TA^{i-1} \end{bmatrix}}_{\in\mathbb{R}^{i\times n}}}_{L_i} \tag{5.38}$$

ist. Dabei sind die Elemente der Matrix C wie folgt definiert:

$$\begin{aligned} c_{1,i} &= 1 \\ c_{1,j} &= \sum_{k=j+1}^{n} c_{1,k} b^T A^{k-(j+1)} a, \quad \text{für } j = i-1, \ldots, 1 \\ c_{p,q} &= c_{p-1,q+1}, \quad \text{für } p = 2, \ldots, i \text{ und } q = 1, \ldots, p \\ c_{p,q} &= 0, \quad \text{für } p = 2, \ldots, i \text{ und } q = p+1, \ldots, i. \end{aligned} \tag{5.39}$$

Beispiel 5.5.2. *Zur Veranschaulichung seien die Matrizen L_2 und L_3 explizit angegeben:*

$$\begin{aligned} L_2 &= \begin{bmatrix} a & Aa \end{bmatrix} \begin{bmatrix} b^T a & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b^T \\ b^T A \end{bmatrix}, \\ L_3 &= \begin{bmatrix} a & Aa & A^2 a \end{bmatrix} \begin{bmatrix} (b^T a)^2 + b^T A a & b^T a & 1 \\ b^T a & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} b^T \\ b^T A \\ b^T A^2 \end{bmatrix}. \end{aligned}$$

Aus Darstellung (5.38) folgt sofort, dass

$$\operatorname{rang}(L_i) \leq i \tag{5.40}$$

ist. Weiterhin gilt für die Summe der Matrizen L_i:

$$\operatorname{rang}\left(\sum_{i=1}^{m} L_i \right) \leq m. \tag{5.41}$$

Aus (5.37),(5.40) und (5.41) ergibt sich der Rang der Summe der Korrekturterme der Matrizen U und V demnach zu

$$\operatorname{rang}(V_k \pm U_{k_1} \pm U_{k_2}) \leq m + 1. \tag{5.42}$$

Damit ist gezeigt, dass aus der Rang-1 Aufdatierung der Matrix A bei Anwendung des Scaling und Squaring Algorithmus eine Korrektur der Matrizen U und V maximal vom Rang $m+1$ stattfindet. An numerischen Beispielen wird in Kapitel 5.6 gezeigt, dass der tatsächlich auftretende Rang noch deutlich geringer als $m+1$ ist.

Berechnung der aufdatierten Matrixexponentialfunktion

In (5.34) und (5.35) sind die Niedrigrangkorrekturen U_{k_1}, U_{k_2} sowie V_k der Matrizen U und V definiert. Aus (5.38) folgt, dass zu deren Berechnung keine Matrix-Matrix Produkte, sondern lediglich Matrix-Vektor Produkte der Form

$$b^T A^i a, \quad A^i a, \quad b^T A^i, \quad i = 0, \ldots, m-1$$

gebildet werden müssen. Demnach ist bis zu dieser Stelle eine Komplexität des Rechenaufwands von $\mathcal{O}(n^2)$ gewährleistet.

Zur Berechnung der aufdatierten Matrixexponentialfunktion ist die Lösung der Matrixgleichung

$$(-U_+ + V_+)e^{A+ab^T} = U_+ + V_+$$

erforderlich. Um auch in diesem Schritt einen quadratischen Aufwand an Operationen beibehalten zu können, ist es nötig, dass die Matrizen U und V sowie die Faktorisierung von $(-U+V)$, welche zur Lösung des Systems $(-U+V)e^A = U+V$ verwendet wurde, bekannt sind. Wurde (5.30) beispielsweise mit Hilfe der LU-Faktorisierung

$$(-U+V) = P_f^T L_f U_f \tag{5.43}$$

gelöst, so ist zur Berechnung von e^{A+ab^T} die Matrixgleichung

$$(P_f^T L_f U_f \underbrace{-U_{k_1} - U_{k_2} + V_k}_{K_r})e^{A+ab^T} = U + V + U_{k_1} + U_{k_2} + V_k \tag{5.44}$$

zu lösen. Da K_r eine Niedrigrangmatrix ist, kann dazu die ursprüngliche Faktorisierung direkt aufdatiert werden. Das heißt mit $\text{rang}(K) = m+1$ ist die dazu nötige Zerlegung

$$(P_f^+)^T L_f^+ U_f^+ = P_f^T L_f U_f + K_r \tag{5.45}$$

beispielsweise als Folge von $m+1$ Rang-1 Aufdatierungen der LU-Faktorisierung möglich. Wie in Kapitel 3 dargestellt wird, ist jede dieser Rang-1 Korrekturen mit $\mathcal{O}(n^2)$ Operationen berechenbar. Die Berechnung der einzelnen Rang-1 Terme kann hierzu zum Beispiel durch eine Singulärwertzerlegung der Matrix C, vgl. (5.39), welche maximal vom Rang $m+1$ ist, erfolgen. Tritt hierbei ein Rangdefekt auf, wird damit die Anzahl der erforderlichen Rang-1 Aufdatierungen sofort reduziert.

Bis zu dieser Stelle sind sämtliche Berechnungen in $\mathcal{O}(n^2)$ Operationen durchführbar. Nur das abschließende s-fache Quadrieren der Lösung zur Rückskalierung erfordert einen kubischen Rechenaufwand. Das bedeutet, ohne Skalierung ist eine Aufdatierung der Matrixexponentialfunktion mit dem hier vorgestellten Verfahren in $\mathcal{O}(n^2)$ möglich.

Der eigentliche Nachteil dieses Vorgehens liegt darin, dass es nicht möglich ist den Skalierungsparameter s für die Aufdatierung anzupassen. Demzufolge muss der Rang-1 Term ab^T vor Berechnung der Korrekturterme U_{k_1}, U_{k_2} sowie V_k mit demselben Skalierungsparameter 2^{-s}, welcher bereits zur Berechnung von e^A verwendet wurde, multipliziert werden. Das bedeutet, dass eine Padé-Approximation der Matrixexponentialfunktion $e^{\frac{A+ab^T}{2^s}}$ durchgeführt wird. Hieraus folgen Probleme hinsichtlich der Approximationsgenauigkeit,

da durch Verwendung des alten Wertes s die Bedingung $\|\frac{1}{2^s}(A+ab^T)\|_1 < \Theta_m$ für Rang-1 Korrekturen mit großer Norm nicht zwangsläufig erfüllt ist. Inwieweit dies die Approximationsgenauigkeit beeinträchtigt, wird anhand numerischer Beispiele in Kapitel 5.6 dargestellt.

Ist zu erwarten, dass die auftretenden Rang-1 Korrekturen eine große Norm besitzen, kann durch eine anfängliche Überskalierung das soeben beschriebene Problem umgangen werden. Dies bedingt allerdings eine größere Anzahl an rechenintensiven Matrix-Matrix Multiplikationen zur abschließenden Rückskalierung.

5.6 Numerische Ergebnisse

5.6.1 Verwendung des Cauchy-Integrals

Die in den Kapiteln 5.3 und 5.5 vorgestellten Verfahren zur Rang-1 Aufdatierung der Matrixexponentialfunktion wurden in MATLAB implementiert. Unter der als im Folgenden exakt bezeichneten Differenz

$$K = e^{A+ab^T} - e^A \tag{5.46}$$

zwischen ursprünglichem und aufdatiertem Wert der Matrix Exponentialfunktion wird der durch die Verwendung der MATLAB Funktion `expm` erhaltene Wert (`expm(A+a*b')-expm(A)`) verstanden. Die Berechnung der Matrixexponentialfunktion durch `expm` wird von MATLAB allerdings mit dem Scaling und Squaring Algorithmus durchgeführt und ist demzufolge kein exakter Wert.

Vergleich von summierter Trapez- und Simpsonregel

In diesem Abschnitt soll das Approximationsverhalten von summierter Trapezregel und summierter Simpsonregel zur Berechnung des Integrals (5.22) gegenübergestellt werden. Hierbei wurde zunächst eine zufällig generierte Diagonalmatrix $A_1 \in \mathbb{R}^{30\times 30}$ mit dem Rang-1 Term $\frac{1}{10}ab^T$ aufdatiert. Die Vektoren $a, b \in \mathbb{R}^{30}$ wurden ebenfalls zufällig generiert. Die Berechnung des Updates der Matrixexponentialfunktion $\tilde{K} = e^{A_1^+} - e^{A_1}$ mit $A_1^+ = A_1 + \frac{1}{10}ab^T$ fand durch die numerische Berechnung des Integrals (5.22) mit der summierte Trapez- sowie summierten Simpsonregel statt. Als Integrationsweg wurde der Kreis $\Gamma : [0, 2\pi] \to \mathbb{C}, t \to m_k + r_k e^{it}$ mit Mittelpunkt $m_k = 0.5$ und Radius $r_k = 2.5$ gewählt. In Abbildung 5.2 sind die Eigenwerte der ursprünglichen- sowie der aufdatierten Matrix sowie die verwendete Integrationskurve dargestellt. Abbildung 5.3 zeigt die Fehlerentwicklung $\|T_N - T_{N-1}\|_2$ beider Integrationsmethoden in Abhängigkeit zur Anzahl

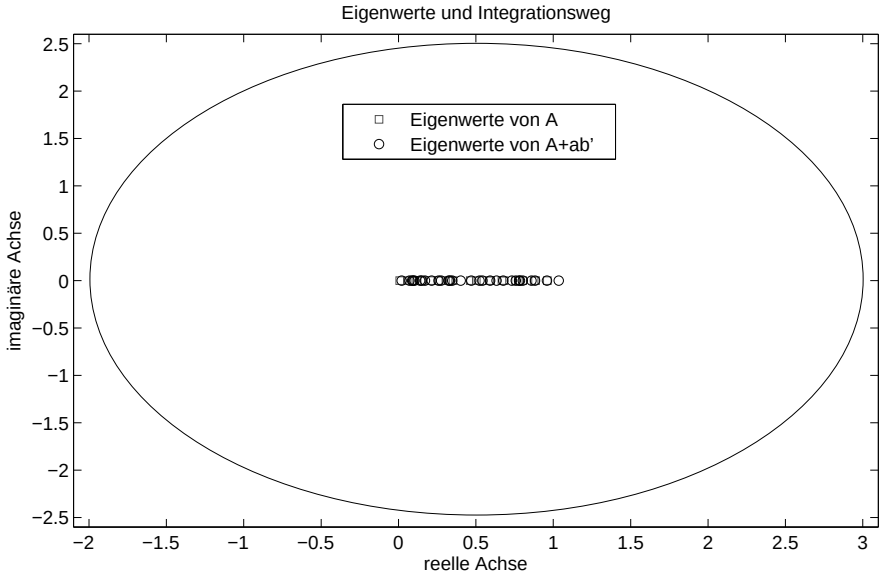


Abbildung 5.2: Eigenwerte und Integrationsweg

der Quadraturpunkte N. Es ist zu sehen, dass in diesem Beispiel die summierte Trapez- und die summierte Simpsonregel einen fast identischen exponentiellen Fehlerabfall aufweisen. Bei einer Stützstellenanzahl ≥ 25 befindet sich dieser Fehler bereits im Bereich der Maschinengenauigkeit. Durch K_T sowie K_S seien die durch summierte Trapez- bzw. summierte Simpsonregel berechneten Differenzen analog zu (5.46) bezeichnet. Die beiden relativen Fehler (bei N=32)

$$\begin{aligned} \frac{\|K_T - K\|_2}{\|K\|_2} &\approx 1.8 \cdot 10^{-15} \\ \frac{\|K_S - K\|_2}{\|K\|_2} &\approx 1.7 \cdot 10^{-15} \end{aligned}$$

sind nahzu identisch. Der numerische Rang der Matrizen K_T bzw. K_S beträgt 5 und ist damit deutlich geringer als der maximal mögliche Wert 16, welcher im Fall reeller Matrizen bei einer Berechneung mit einer Stützstellenzahl von $N = 32$ auftreten kann.

Der Vergleich beider Integrationsverfahren soll am Beipiel einer zweiten, in diesem Fall vollbesetzten und ebenfalls zufällig erzeugten Matrix $A_2 \in \mathbb{R}^{30\times 30}$ wiederholt werden. Zur Aufdatierung $A_2^+ = A_2 + ab^T$ wurden erneut zwei zufällig generierte Vektoren verwendet. Als Integrationsweg wurde wiederum ein Kreis mit Radius $r_k = 18$ und Mittelpunkt $m_k = 12$ gewählt. Darstellung 5.4 zeigt die Eigenwerte der Matrizen A_2 und A_2^+ sowie den Integrationsweg Γ. In Abbildung 5.5 wird die Fehlerentwicklung bei Verwendung der summierten Trapez- bzw. der summierten Simpsonregel dargestellt. Erneut

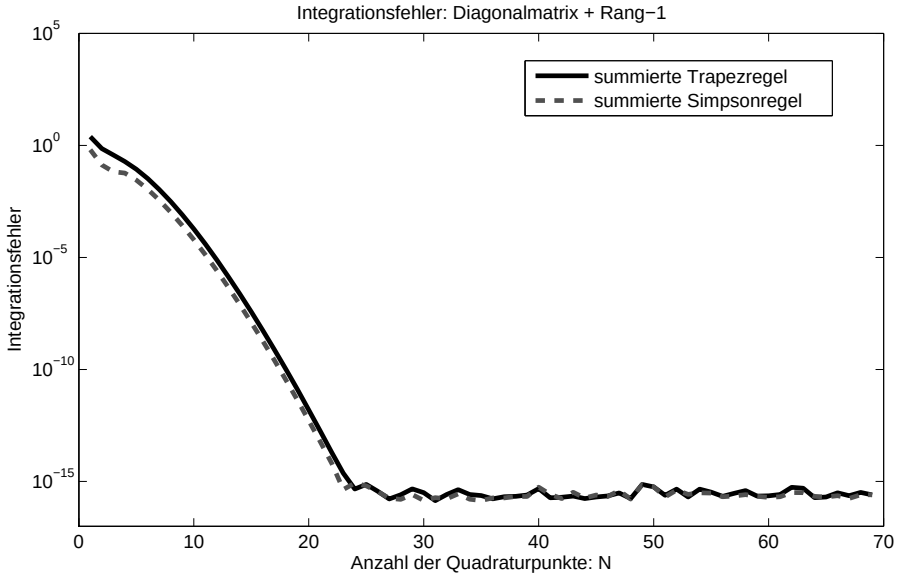


Abbildung 5.3: Fehlerentwicklung von summierter Trapez- und summierter Simpsonregel

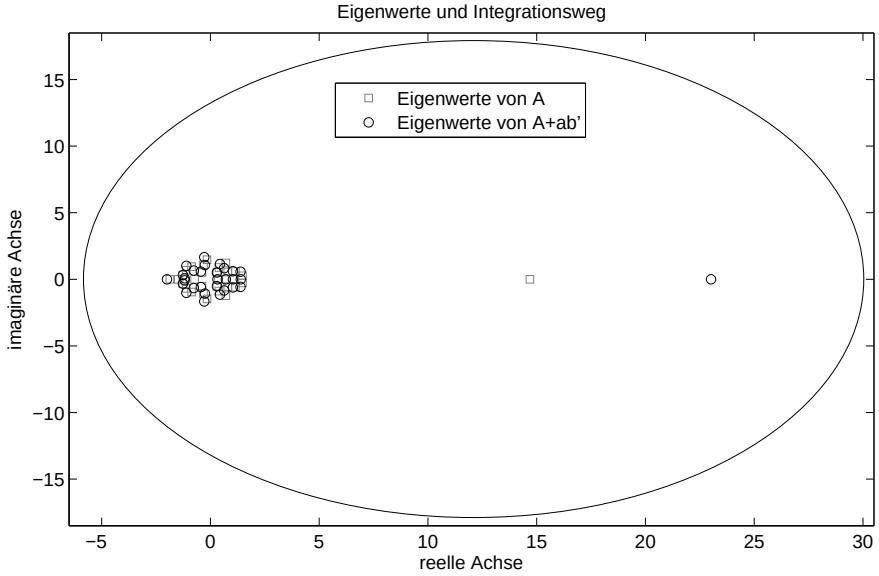


Abbildung 5.4: Eigenwerte und Integrationsweg

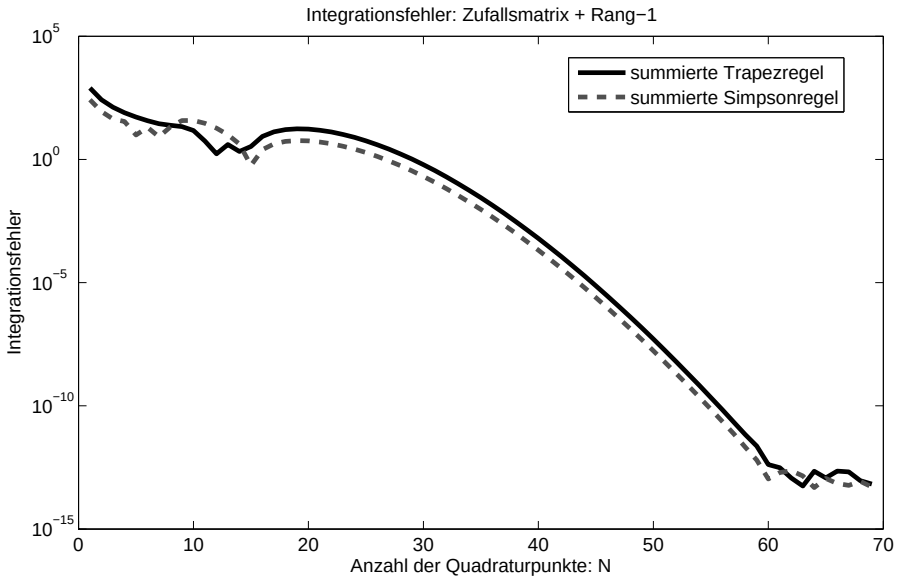


Abbildung 5.5: Fehlerentwicklung von summierter Trapez- und summierter Simpsonregel

haben beide Integrationsverfahren eine fast identische Fehlerentwicklung. Aufgrund der Breite des Spektrums von A_2^+ sowie dem größten Eigenwert $\lambda_{max} \approx 23$ von A_2^+ enthält die Matrix $e^{A_2^+}$ extrem große Werte und hat eine Konditionszahl von $\kappa(e^{A_2^+}) \approx 9.8 \cdot 10^{10}$. Dennoch erreichen beide Verfahren (bei $N = 64$) relative Fehler von

$$\begin{aligned} \frac{\|K_T - K\|_2}{\|K\|_2} &\approx 4.5 \cdot 10^{-14} \\ \frac{\|K_S - K\|_2}{\|K\|_2} &\approx 4.9 \cdot 10^{-14}. \end{aligned}$$

In diesem Beispiel ist allerdings eine deutlich größere Anzahl an Quadraturpunkten nötig, um diese Approximationsgenauigkeit zu erreichen. Der numerische Rang der berechneten Korrekturen K_S und K_T ist in diesem Beispiel erneut, unabhängig von der verwendeten Stützstellenzahl, gleich 5.

Hiermit konnte die exponentielle Fehlerentwicklung der numerischen Integration des zur Berechnung der aufdatierten Matrix Exponentialfunktion benötigten Integrals anhand zweier Beispiele belegt werden. Es ist zu sehen, dass aufgrund dieser Fehlerentwicklung auch die Verwendung von Verfahren höherer Ordung keinen entscheidenden Vorteil gegenüber der summierten Trapezregel bietet. Des Weiteren ist zu erkennen, dass in beiden Fällen der numerische Rang des Aufdatierungsterms K deutlich geringer ist, als es bei der Berechnung mit der jeweils maximal verwendeten Stützstellenzahl möglich wäre.

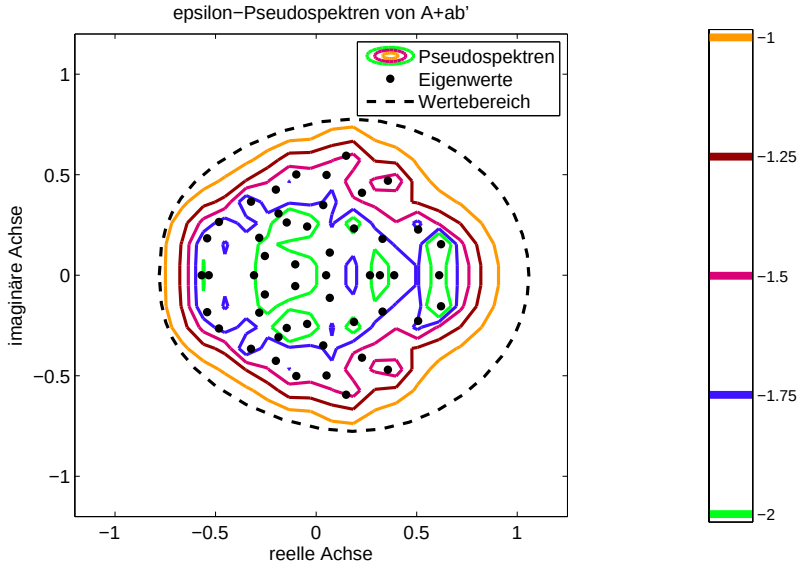


Abbildung 5.6: ε-Pseudospektren von $A + ab^T$

Variation des Integrationskreises

In den nächsten Beispielen soll dargestellt werden, wie sich die Veränderung von Mittelpunkt und Radius des Integrationskreises auf das Approximationsverhalten der summierten Trapezregel auswirkt. Hierfür wurde die Matrix $A \in \mathbb{R}^{50\times 50}$ als MATLAB Testmatrix der Art randsvd gewählt. Dabei handelt es sich um eine Zufallsmatrix, deren Eigenwerte um den Ursprung der komplexen Ebene gleichverteilt sind. Als Kondition der Matrix wurde $\kappa(A) = 100$ gewählt. Der Rang-1 Term zur Aufdatierung der Matrix A wurde aus zwei zufällig erzeugten, normierten Vektoren $a, b \in \mathbb{R}^{50}$ gebildet. In Abbildung 5.6 sind verschiedene ε-Pseudospektren von A dargestellt. Die unterschiedlichen Kurven geben hierbei die ε-Pseudospektren zu den Werten $\varepsilon = 10^x$ mit $x = -1, -1.25, -1.5, -1.75, -2$ an. Diese wurden mit dem Paket EIGTOOL generiert. Der Wert ε^{-1} wächst somit von innen nach außen. Die Grafik verdeutlicht, dass der Wertebereich in diesem Fall eine gute Approximation an das Pseudospekrum zum Wert $\varepsilon = 10^{-2}$ ist. Abbildung 5.7 zeigt die Norm der Resolventen $\|(zI - (A + ab^T))^{-1}\|_2$. Diese Darstellung zeigt, dass die Norm im Bereich der Eigenwerte von $A + ab^T$ sehr stark wächst. Aus diesem Grund ist ein Integrationsweg, welcher zu dicht am Spektrum von $A + ab^T$ liegt, ungeeignet. Zur Berechnung von $K_T = e^{A+ab^T} - e^A$ wird in einem ersten Beispiel der Radius des Kreises Γ variiert. Der Mittelpunkt von Γ wurde aus der äußeren Hülle der Wertebereiche von A sowie $A + ab^T$ bestimmt. Zu deren Berechnung wurde der Algorithmus aus [51] ver-

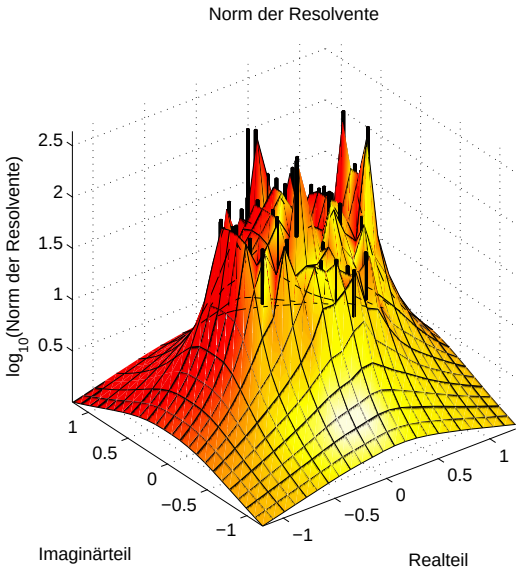


Abbildung 5.7: Norm der Resolventen $\|(zI - (A + ab^T))^{-1}\|_2$

wendet. Grafik 5.8 zeigt die verschiedenen Radien, welche zur Berechnung des Integrals verwendet wurden. In Abbildung 5.9 sind die zugehörigen Fehlerentwicklungen bei der numerischen Integration in Abhängigkeit zur Stützstellenzahl dargestellt. Der Radius r_0 des innersten Kreises wurde unter der Bedingung, dass Γ_{r_0} die Wertebereiche von A und $A + ab^T$ umschließt, kleinstmöglich gewählt. Für die weiteren Kreise wurde der Radius um jeweils eins erhöht. Abbildung 5.9 zeigt, dass mit dem Radius r_0 das schlechteste Approximationsverhalten der summierten Trapezregel erzielt wird. Da in diesem Fall der Abstand der Kurve zu den Spektren von A bzw. $A + ab^T$ gering ist, kann c in (5.23) nur sehr klein gewählt werden, da die Norm der Resolventen, siehe Abbildung 5.7, ansonsten sehr groß wird. Dies bedingt die schlechte Fehlerentwicklung für den Radius r_0.

Das beste Fehlerverhalten ergibt sich für den Radius r_2. Für größere Werte lässt die Approximationsgenauigkeit wieder nach. Dies liegt am Term $r_k e^{R_+ + c}$ aus 5.23, welcher mit zunehmendem Radius wächst und damit eine höhere Stützstellenzahl zur Integration erfordert.

Im zweiten Beispiel wird der Radius des Integrationskreises konstant als r_2 festgelegt. Stattdessen wird der Mittelpunkt des Kreises variiert. Abbildung 5.10 zeigt die verschiedenen Kreise, welche zur Integration verwendet wurden. Hierbei wurde m_0 als der Schwerpunkt der Vereinigung der Wertebereiche von A und $A + ab^T$ bestimmt. Die weiteren Werte wurden jeweils durch eine Erhöhung bzw. Reduktion von m_0 um 0.5 sowie 1 gewählt. In Abbildung 5.11 ist zu sehen, dass der Kreis mit Mittelpunkt m_0 klar das

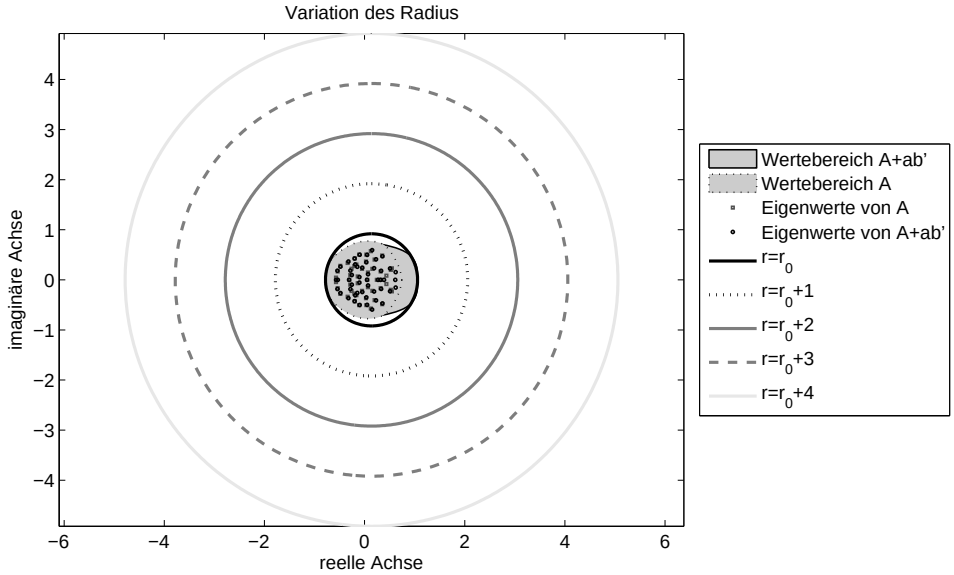


Abbildung 5.8: Variation des Radius des Integrationskreises

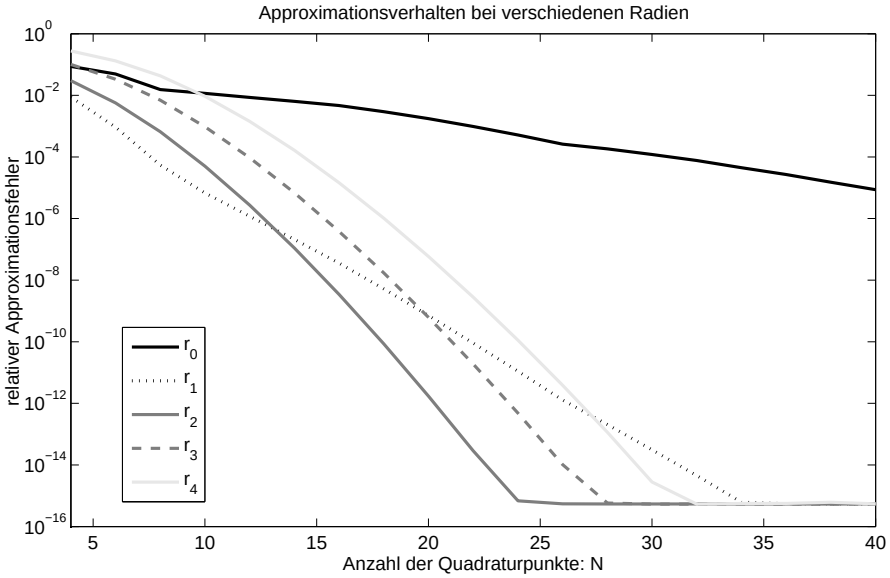


Abbildung 5.9: Fehlerverhalten bei Variation des Radius

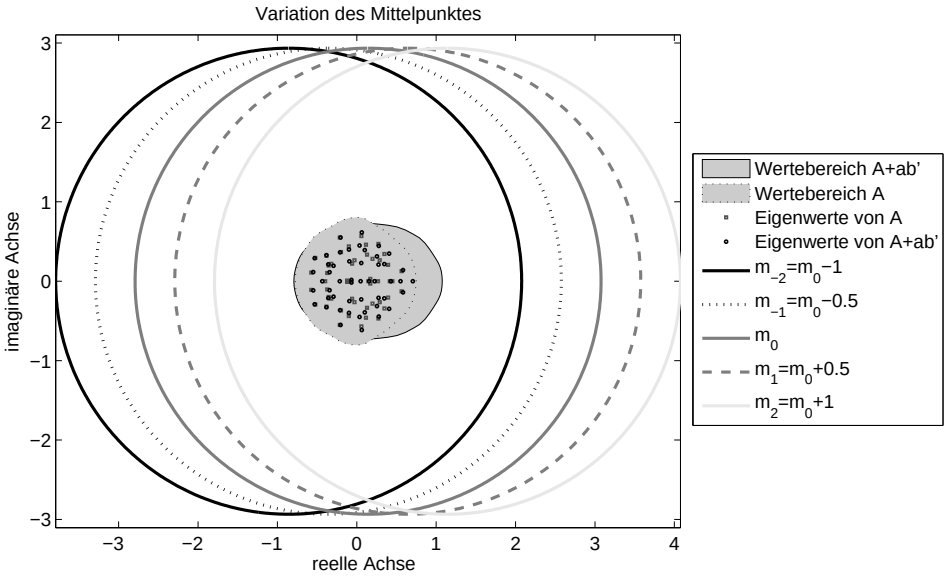


Abbildung 5.10: Variation des Mittelpunktes des Integrationskreises

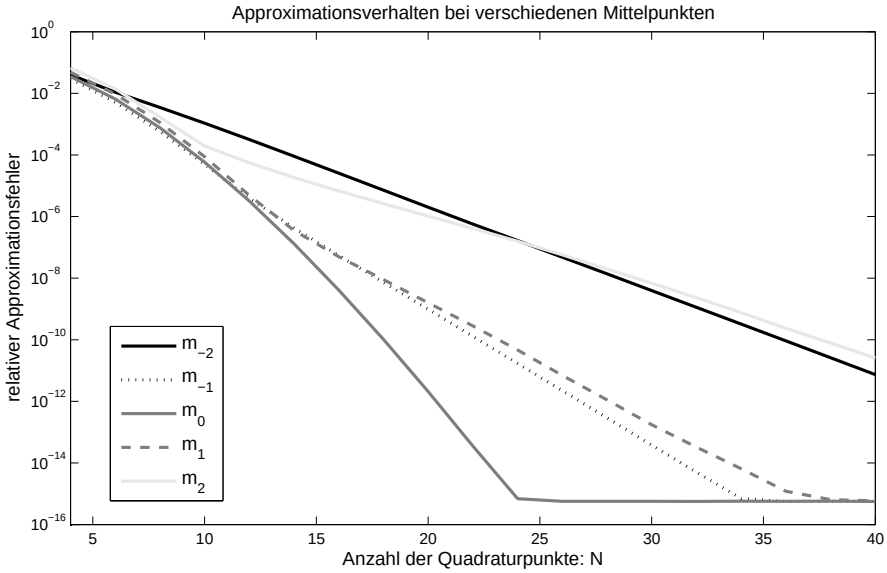


Abbildung 5.11: Fehlerverhalten bei Variation des Mittelpunktes

Typ	Dim.	N	Genauigkeit ϵ	rang(K_T)	$\kappa(A)$	$\kappa(A+ab^T)$	Radius	Mittelp.
Poisson	100	40	$6.1 \cdot 10^{-14}$	9	48.3	19.8	7.68	4
Hankel	100	48	$4.0 \cdot 10^{-15}$	5	3.3	2.2	3.4	0.05
svdrand	1000	40	$2.8 \cdot 10^{-14}$	7	100	165	2.31	-0.15
Toeplitz	500	96	$3.6 \cdot 10^{-14}$	9	3524	3612	16.1	-0.11
random$_{-10}$	250	50	$3.1 \cdot 10^{-15}$	6	25021	21800	6.57	-5.92
diag	999	50	$6 \cdot 10^{-15}$	3	10000	13390	10.6	-4.73

Tabelle 5.3: Rang-1 Aufdatierung der Matrixexponentialfunktion unterschiedlicher Matrizen

beste Fehlerverhalten liefert. Schon leichte Abweichungen in positive wie negative Richtung sorgen für eine deutlich schlechtere Approximation des Integrals. Dieses Verhalten ist analog zur vorherigen Betrachtung durch (5.23) und (5.24) begründet.

Aufdatierung verschiedener Matrizen

In diesem Abschnitt sollen einige Ergebnisse der Aufdatierung der Matrixexponentialfunktion für verschiedene Matrizen gezeigt werden. Während zahlreicher numerischer Tests hat sich herausgestellt, dass in den meisten Fällen der Schwerpunkt der Vereinigung der Wertebereiche von A und $A + ab^T$ als Kreismittelpunkt sowie ein um den Faktor zwei gestreckter Radius des Wertebereichs als Radius des Integrationskreises geeignet sind. Die in Tabelle 5.3 dargestellten Ergebnisse wurden mit dieser Wahl erzielt. Die Rang-1 Terme wurden jeweils aus den zufällig generierten, normierten Vektoren $a, b \in \mathbb{R}^n$ gebildet. In der ersten Spalte ist der Typ der Matrix angegeben. Die Matrizen 'Poisson', 'Hankel', 'svdrand' und 'Toeplitz' wurden dabei mit der MATLAB Funktion gallery erzeugt. 'random$_{-10}$' bezeichnet eine mit dem MATLAB rand Befehl generierte und mit -10^{-1} skalierte Zufallsmatrix. Die Diagonalmatrix in der letzten Zeile wurde so gewählt, dass jeweils gleichhäufig die Diagonaleinträge -0.001, -0.1 sowie -10 auftreten. In der zweiten Spalte ist die Dimension der jeweiligen Matrizen angegeben.

Zur Aufdatierung wurde die Stützstellenanzahl bei 16 beginnend jeweils solange um 2 erhöht, bis zwei aufeinanderfolgende Approximationen des Integrals numerisch identisch waren. Die dazu jeweils benötigte Stützstellenzahl wird in der dritten Spalte gezeigt. In der vierten Spalte ist der dabei auftretende relative Fehler der berechneten Niedrigrangkorrektur K_T im Vergleich zur mit MATLAB berechneten exakten Lösung $K = e^{A+ab^T} - e^A$ angegeben, d.h. $\epsilon = \frac{\|K_T - K\|_2}{\|K\|_2}$. Die Spalte rang$(K_T)$ gibt an, welchen

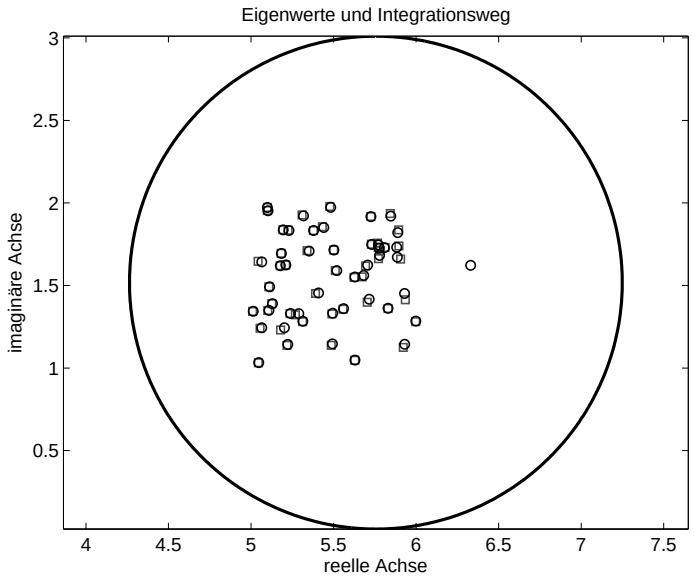


Abbildung 5.12: Eigenwerte und Integrationsweg

numerischen Rang die additive Korrektur zwischen ursprünglicher und aufdatierter Matrixexponentialfunktion besitzt. In den folgenden Spalten sind die Konditionszahlen der Matrizen sowie Mittelpunkt und Radius des verwendeten Integrationskreises gegeben. Es ist zu erkennen, dass für sämtliche getestete Beispiele eine sehr gute Genauigkeit bei moderater Stützstellenanzahl erzielt werden konnte. Des Weiteren ist zu sehen, dass der numerische Rang des Korrekturterms zwischen altem und aufdatiertem Wert deutlich geringer als das in der jeweiligen Berechnung durch die zugehörige Anzahl an Quadraturpunkten mögliche Maximum N ist.

Anwendung auf andere Funktionen

Die Aufdatierung mit Hilfe des Cauchy-Integrals wurde abschließend für weitere Matrixfunktionen getestet. Hierbei wurde die Ausgangsmatrix der Dimension 50×50 und der Rang-1 Term so gewählt, dass sowohl die Eigenwerte von A als auch von $A + ab^T$ ausschließlich positive Realteile besitzen. Bei der Wahl des Integrationskreises ist weiterhin die Singularität auf $(-\infty, 0]$ für den Hauptzweig des Logarithmus sowie für die auf diesen zurückführbare Wurzelfunktion zu beachten. Die Spektren der Matrizen sowie der Integrationsweg sind in Abbildung 5.12 zu sehen. Die auftretende Fehlerentwicklung bei der Integration ist für einige ausgewählte Funktionen in Abbildung 5.13 dargestellt. Es ist zu sehen, dass die Winkelfunktionen, welche sich auch als Summe von Exponential-

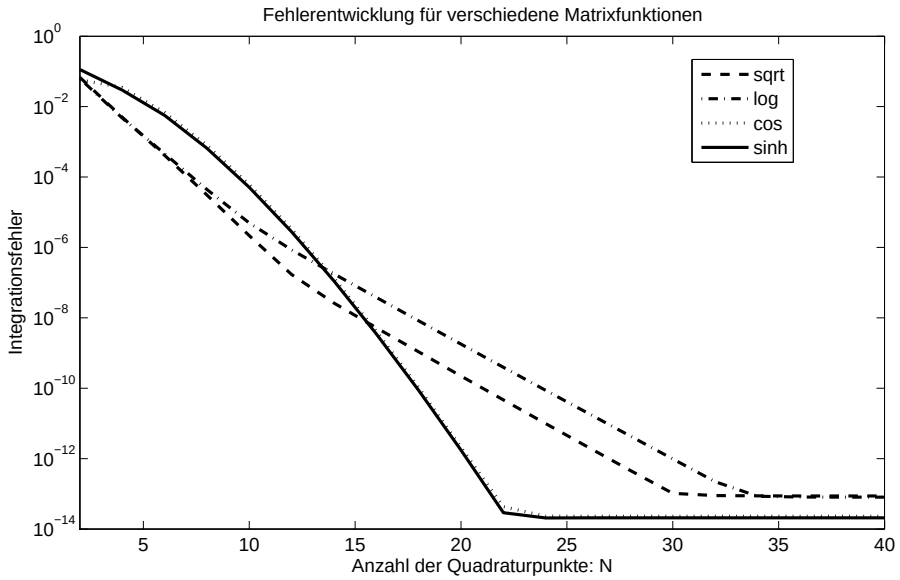


Abbildung 5.13: Fehlerentwicklung für verschiedene Matrixfunktionen

funktionen darstellen lassen, eine analoge Fehlerentwicklung wie die Exponentialfunktion selbst aufweisen. Das Fehlerverhalten für die Quadratwurzel sowie die Logarithmusfunktion ist etwas schlechter. Dennoch liegt auch in diesen Fällen eine exponentiell abfallende Fehlerentwicklung vor.

5.6.2 Verwendung von Scaling und Squaring

Das auf dem Scaling und Squaring Verfahren beruhende Vorgehen zur Aufdatierung der Matrixexponentialfunktion wurde für verschiedene Matrizen getestet. Im Gegensatz zum zuvor vorgestellten Verfahren wird dabei der Korrekturterm K, vgl. (5.3), nicht explizit berechnet. Stattdessen wird die Rang-1 Korrektur und die bereits mit dem Scaling und Squaring Verfahren zuvor berechnete Matrixexponentialfunktion e^A genutzt, um den aufdatierten Funktionswert effizient zu berechnen. Hierzu ist es nötig, dass die Matrizen U, V, die Faktorisierung von $(-U+V)$, siehe (5.43), sowie der Skalierungsparameter s aus der vorherigen Berechnung von e^A vorliegen.

Die beiden Diagramme in Abbildung 5.14 zeigen die Abhängigkeit der Lösungsgenauigkeit von der Anzahl der Skalierungsschritte s. Die linke Grafik stellt den relativen Fehler

$$\varepsilon = \frac{\|\tilde{E} - e^{A+ab^T}\|_2}{\|e^{A+ab^T}\|_2} \tag{5.47}$$

der berechneten Lösung $\tilde{E} = e^{A+ab^T}$ im Vergleich zum exakten mit MATLAB berechneten

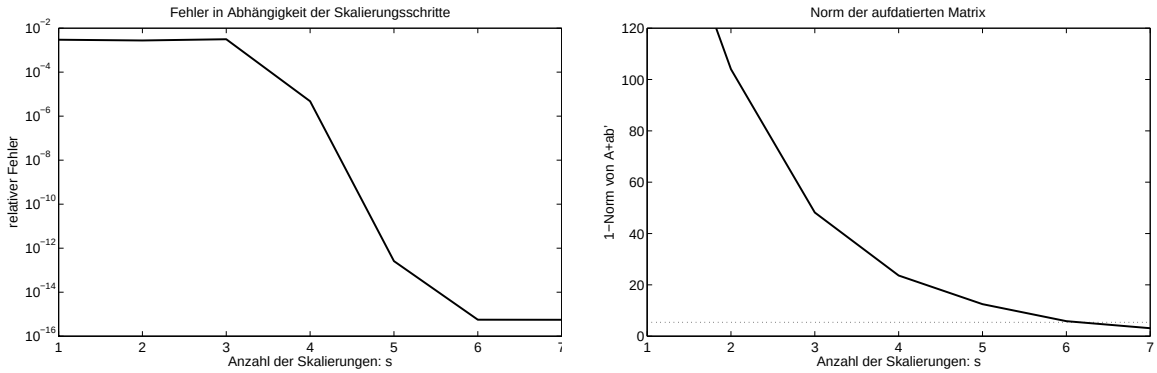


Abbildung 5.14: Approximationsfehler und Norm in Abhägigkeit der Skalierungsschritte

		$s=0$		$s=1$		$s=2$		$s=3$	
Typ	Dim.	$\|A_+\|_1^s$	ϵ_0	$\|A_+\|_1^s$	ϵ_1	$\|A_+\|_1^s$	ϵ_2	$\|A_+\|_1^s$	ϵ_3
Poisson	100	9.49	$8.0 \cdot 10^{-13}$	4.74	$*1.4 \cdot 10^{-15}$	2.37	$1.2 \cdot 10^{-15}$	1.18	$3.4 \cdot 10^{-15}$
Hankel	100	6.43	$1.0 \cdot 10^{-15}$	3.22	$*4.6 \cdot 10^{-16}$	1.61	$1.7 \cdot 10^{-15}$	0.8	$2.5 \cdot 10^{-15}$
svdrand	1000	25.8	$3.7 \cdot 10^{-15}$	12.9	$4.9 \cdot 10^{-15}$	6.45	$6.6 \cdot 10^{-16}$	3.23	$*4.4 \cdot 10^{-16}$
Toeplitz	500	21.2	0.002	10.6	$9.9 \cdot 10^{-11}$	5.28	$*2.6 \cdot 10^{-18}$	2.64	$6.2 \cdot 10^{-16}$
$-\text{random}_{10}$	250	13.2	$5.3 \cdot 10^{-14}$	6.6	$3.7 \cdot 10^{-15}$	3.3	$*3.4 \cdot 10^{-15}$	1.65	$1.0 \cdot 10^{-14}$
diag	1000	11.4	$5.7 \cdot 10^{-13}$	5.7	$*2.8 \cdot 10^{-15}$	2.85	$3.4 \cdot 10^{-15}$	1.43	$5.9 \cdot 10^{-15}$

Tabelle 5.4: Rang-1 Aufdatierung verschiedener Matrizen, $\|ab^T\|_1 = 1$

Wert dar. Die x-Achse gibt hierbei die Anzahl der verwendeten Skalierungsschritte an. In der rechten Grafik sind die zugehörigen Normen $\|A_+\|_1^s = \|\frac{1}{2^s}(A + ab^T)\|_1$ der aufdatierten und gemäß s skalierten Matrizen gegeben. Es ist zu sehen, dass der Fehler ϵ bei 6 Skalierungsschritten in den Bereich der Maschinengenauigkeit fällt. In diesem Fall ist die Norm der aufdatierten und skalierten Matrix gleich 5.8, was der theoretischen Schranke $\Theta_{13} = 5.4$ nahe kommt.

Das Scaling und Squaring Aufdatierungsverfahren wurde weiterhin auf die in Tabelle 5.3 verwendeten Matrizen angewendet. Die dabei für zufällig generierte, normierte Rang-1 Aufdatierungen erzielten Ergebnisse sind in Tabelle 5.4 dargestellt. Neben Typ und Dimension der Matrizen sind zu verschiedenen Skalierungsparametern die Norm der aufdatierten, skalierten Matrix sowie die mit diesem Skalierungsparameter erreichte relative Genauigkeit ϵ_i, vgl. (5.47) gegeben. Die mit * gekennzeichneten Werte markieren diejenigen Ergebnisse, welche erzielt wurden, wenn s so gewählt ist, dass die Berechnung von e^A im Rahmen der Maschinengenauigkeit berechnet werden konnte, d.h. $\|2^{-s}A\|_1 \leq \Theta_{13}$. Da die Norm des Rang-1 Korrekturterms in diesen Beispielen gering ist, werden mit dieser Wahl von s ebenfalls sehr genaue Ergebnisse der Matrixexponentialfunktion e^{A+ab^T} erzielt. Des Weiteren ist zu erkennen, dass eine leichte Überskalierung ebenfalls noch

		$s=0$		$s=1$		$s=2$		$s=3$	
Typ	Dim.	$\|A_+\|_1^s$	ε_0	$\|A_+\|_1^s$	ε_1	$\|A_+\|_1^s$	ε_2	$\|A_+\|_1^s$	ε_3
Poisson	100	41.9	0.04	20.9	$*1.7 \cdot 10^{-9}$	10.5	$6.6 \cdot 10^{-16}$	5.2	$2.6 \cdot 10^{-16}$
Hankel	100	39.3	0.03	19.6	$*3.3 \cdot 10^{-11}$	9.8	$1.9 \cdot 10^{-16}$	4.9	$2.6 \cdot 10^{-16}$
svdrand	1000	44.6	0.003	22.3	$8.9 \cdot 10^{-12}$	11.1	$1.8 \cdot 10^{-16}$	5.6	$*2.5 \cdot 10^{-17}$
Toeplitz	500	57.7	0.004	28.9	0.0039	14.4	$*2.7 \cdot 10^{-14}$	7.2	$4.7 \cdot 10^{-17}$
$-\text{random}_{10}$	250	27.2	$6.5 \cdot 10^{-15}$	13.6	$9.3 \cdot 10^{-17}$	6.8	$*9.0 \cdot 10^{-17}$	3.4	$1.0 \cdot 10^{-16}$
diag	1000	46.1	0.0002	23.1	$*3.0 \cdot 10^{-13}$	11.5	$2.3 \cdot 10^{-17}$	5.8	$2.5 \cdot 10^{-17}$

Tabelle 5.5: Rang-1 Aufdatierung verschiedener Matrizen, $\|ab^T\|_1 = 25$

zu einer sehr hohen Genauigkeit der Lösung führt. Wird dahingegen s zu groß gewählt, so wird zum einen der benötigte Rechenaufwand beim abschließenden Quadrieren sehr groß. Zum anderen ist diese Rückskalierung ein numerisch kritischer Punkt im Scaling und Squaring Algorithmus, siehe [46]. Das bedeutet je größer s ist, umso größer ist auch der Einfluss von Rundungsfehlern auf das Ergebnis. Hieraus folgt, wenn vor der Berechnung des Ausgangswertes e^A bereits bekannt ist, dass die Normen der Aufdatierungen klein im Verhältnis zur Norm von A sind, so kann s wie im originalen Scaling und Squaring Verfahren gewählt werden. Auch die Wahl von $s+1$ ist geeignet. In diesem Fall ist zwar eine Matrix-Matrix Multiplikation beim abschließenden Quadrieren mehr zu berechnen, dafür bietet diese leichte Überskalierung einen Freiraum für eventuelle Rang-1 Terme größerer Norm.

Im zweiten Beispiel wurden dieselben Matrizen mit Rang-1 Korrekturen der Norm $\|ab^T\|_1 = 25$ modifiziert. Tabelle 5.5 ist zu entnehmen, dass in diesem Fall die Wahl eines geeigneten Skalierungsparameters problematisch ist. Da die Norm des Aufdatierungsterms in diesem Fall einen größeren Einfluss auf die Genauigkeit der Lösung hat, müsste diese in die Wahl des geeigneten Skalierungsparameters einbezogen werden. Dies setzt allerdings voraus, dass der Term ab^T bereits bei der Berechnung von e^A bekannt ist. Eine zu kleine Wahl von s führt zu sehr großen Ungenauigkeiten der Lösung, wie beispielsweise an der Poisson Matrix zu sehen ist. Wird hingegen der Skalierungsparameter zu groß gewählt führt dies bereits bei der Berechnung von e^A zu erhöhtem Berechnungsaufwand sowie zu numerischen Stabilitätsproblemen.

Für diesen Fall bietet sich als Alternative die Aufdatierung mit Hilfe des Cauchy-Integrals an, siehe Kapitel 5.3. Für weitere, eventuell darauf folgende Aufdatierungen steht nach Durchführung dieses Verfahrens allerdings die Faktorisierung der aktuellen Matrix $(-U+V)$ nicht mehr zur Verfügung. Eine weitere Aufdatierung mit Hilfe des Scaling und Squaring Verfahrens ist demnach erst wieder nach einer expliziten Neuberechnung der aktuellen Matrixexponentialfunktion mit diesem möglich.

5.6.3 Anwendungsbeispiel

In diesem Abschnitt soll die Niedrigrangaufdatierung der Matrixexponentialfunktion in einem Beispiel der Netzwerktheorie angewendet werden, vgl. [19]. Es sei $A \in \mathbb{R}^{n\times n}$ die symmetrische Adjazenzmatrix eines Netzwerks, deren Einträge $a_{ij} = 1$ für $i, j = 1, \ldots, n$ sind, falls die Knoten i und j direkt miteinander verbunden sind und Null sonst. Weiterhin sei $a_{ii} = 0$ für alle $i = 1, \ldots, n$.

Zwei Begriffe in der Analyse von Netzwerken sind centrality (Zentralität) sowie communicability (Kommunizierbarkeit). Centrality lässt sich als Maß auffassen, wie stark ein Knoten in ein Netzwerk eingebunden ist. Zur Veranschaulichung dieses Wertes soll folgende Betrachtung durchgeführt werden. Durch $(A^2)_{ii} = \sum_{k=1}^{n} a_{ik}a_{ki}$ ist die Anzahl der Kanten gegeben, welche mit dem Knoten i verbunden sind. Nach einer weiteren Multiplikation mit A, gibt der Wert $(A^3)_{ii}$ die Anzahl an Wegen vom Knoten i über zwei weitere Knoten zu sich selbst zurück an, d.h. $i \to k_1 \to k_2 \to i$. Jede weitere Potenzierung $(A^m)_{ii}$ liefert die Anzahl geschlossener Wege der Länge m vom/zum Knoten i. Nach Aufsummierung und Gewichtung dieser Werte ergibt sich aus

$$\begin{aligned} C_i &= (I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots)_{ii} \\ &= (e^A)_{ii} \end{aligned}$$

die subgraph centrality des Knotens i, vgl. [20]. Dieser Wert wird auch als Estrada Index bezeichnet [40].

Die communicability zweier Knoten i und j ergibt sich auf analoge Weise durch $\tilde{C}_{ij} = (e^A)_{ij}$, vgl. [18].

In folgendem Beispiel soll angenommen werden, dass die Adjazenzmatrix A eines Netzwerks, sowie deren Exponential e^A gegeben sind. Ziel ist die Berechnung der Matrixexponentialfunktion der zum Netzwerk zugehörigen Adjazenzmatrix A_+, nachdem eine Kante in diesem entfernt bzw. hinzugefügt wurde.

Wird im Netzwerk eine Verbindung zwischen zwei Knoten i und j entfernt, so müssen in der zugehörigen Adjazenzmatrix A die Werte a_{ij} und a_{ji} Null gesetzt werden. Dies entspricht der Rang-2 Aufdatierung

$$A_+ = A - UV^T,$$

mit $U = \begin{bmatrix} e_i & e_j \end{bmatrix}$ und $V = \begin{bmatrix} e_j & e_i \end{bmatrix}$, wobei e_i und e_j den i-ten bzw. j-ten Einheitsvektor bezeichnen. Das Hinzufügen einer Kante entspricht analog der Addition $A_+ = A + UV^T$.

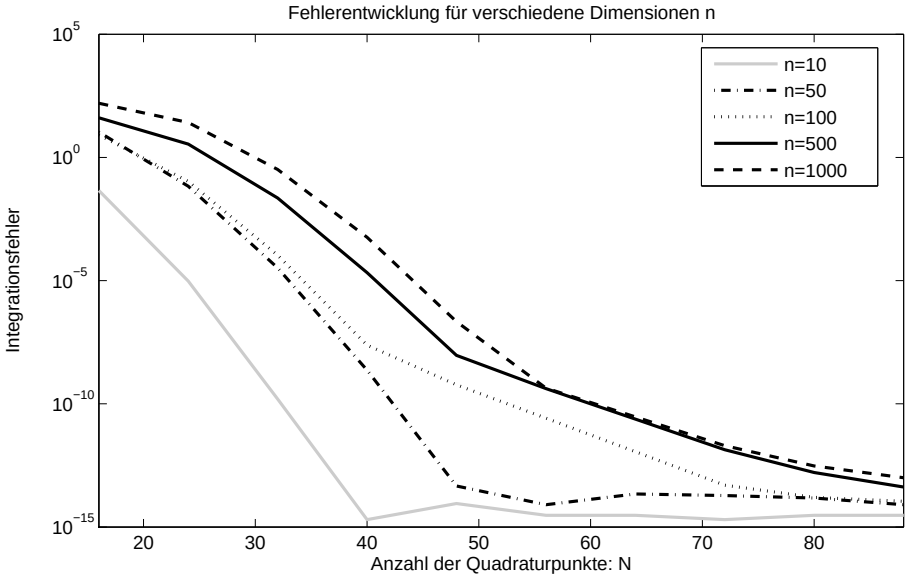


Abbildung 5.15: Fehler des Korrekturterms K bei Rang-2 Aufdatierung der Adjazenzmatrix

Im Folgenden werden in Adjazenzmatrizen unterschiedlicher Dimension, welche mit Hilfe der CONTEST-Sammlung[1] an Testmatrizen erzeugt wurden, jeweils einzelne Kanten entfernt. Die Änderung der zugehörigen Matrixexponentialfunktion $K = e^{A_+} - e^A$ soll dazu mit Hilfe des Cauchy-Integrals berechnet werden.

Diese Rang-2 Aufdatierung kann analog zum in Kapitel 5.3 beschriebenen Vorgehen durchgeführt werden. In (5.6) werden dazu die Vektoren a und b durch U und V ersetzt. Somit ergibt sich das Integral

$$e^{A_+} = e^A + \frac{1}{2\pi i} \int_\Gamma e^z (zI - A)^{-1} U (I - V^T (zI - A)^{-1} U)^{-1} V^T (zI - A)^{-1} \, \mathrm{d}z \,.$$

Die numerische Integration erfolgt erneut mit der summierten Trapezregel. Als Kurve Γ wurde jeweils der Kreis mit Mittelpunkt $m = \lambda_{\min}(A) + 0.5(\lambda_{\max}(A) - \lambda_{\min}(A))$ und Radius $r = (\lambda_{\max}(A) - \lambda_{\min}(A))$ gewählt. Abbildung 5.15 zeigt die Fehlerentwicklung des berechneten Korrekturterms K an Problemen der Dimension $n = 10, 50, 100, 500, 1000$ zu ansteigender Anzahl an Quadraturpunkten N. Es ist zu sehen, dass für alle betrachteten Beispiele, auch im Fall der hier vorliegenden Rang-2 Aufdatierung, eine sehr gute Genauigkeit der Lösung erzielt werden kann. Die Anzahl der benötigten Stützstellen steigt hierbei für größer werdende Dimensionen an. In Abbildung 5.16 sowie Tabelle

[1] http://www.mathstat.strath.ac.uk/research/groups/numerical_analysis/contest

i	$n = 50$	$n = 100$	$n = 500$
1	2105.177846505	1352.737593520	5771.636744320
2	223.7470211314	446.9410950297	3936.121210891
3	5.561248229304	21.48289145399	23.13651335762
4	2.506648929305	5.151579147835	12.52309859087
5	0.675735238817	2.167234811347	5.878826404561
6	0.080959972668	0.254880185091	0.637675865843
7	0.046447771286	0.091984837746	0.322314180145
8	0.002289466523	0.009463422050	0.037269434205
9	0.000955219496	0.002075672069	0.011709496704
10	0.000045957882	0.000216746566	0.001412711232

Tabelle 5.6: Die größten 10 Singulärwerte $\sigma_i(K)$

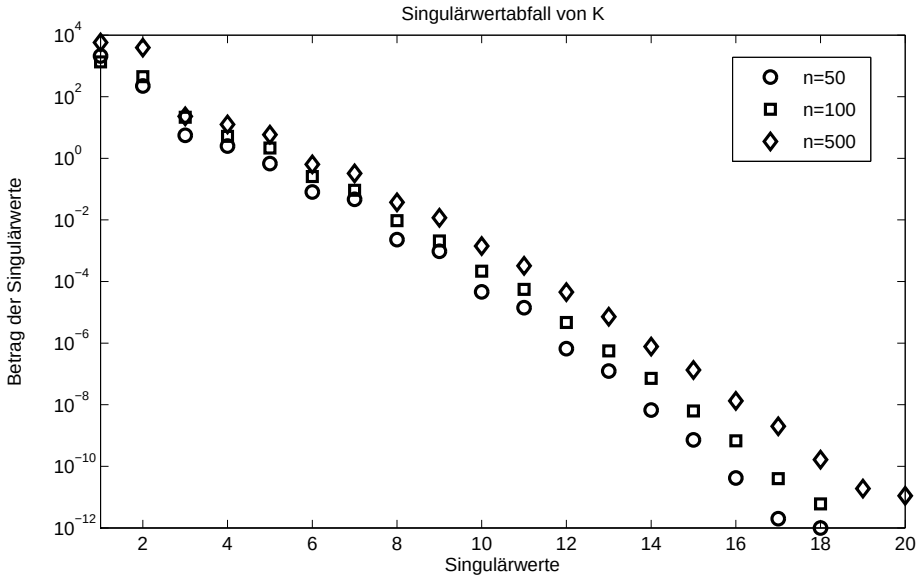


Abbildung 5.16: Die größten 20 Singulärwerte von K

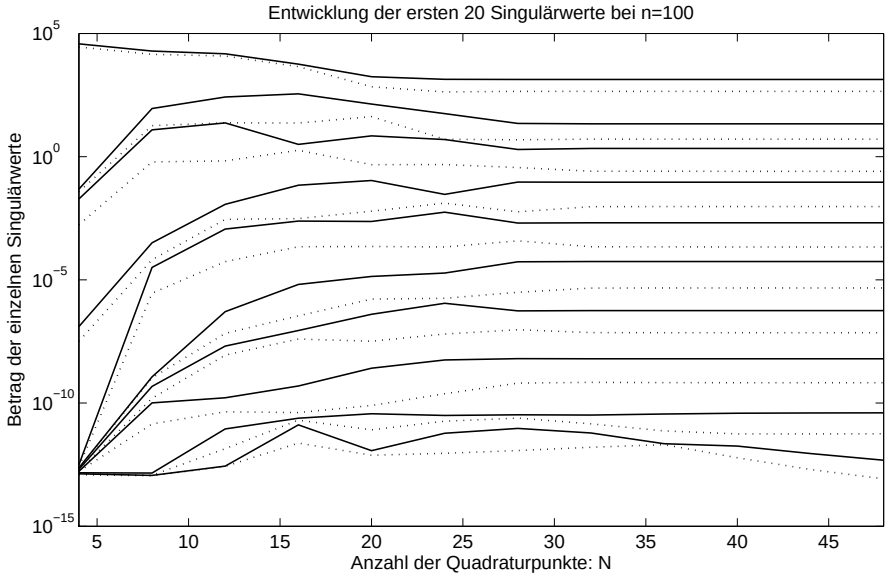


Abbildung 5.17: Entwicklung der größten Singulärwerte von K, $n = 100$

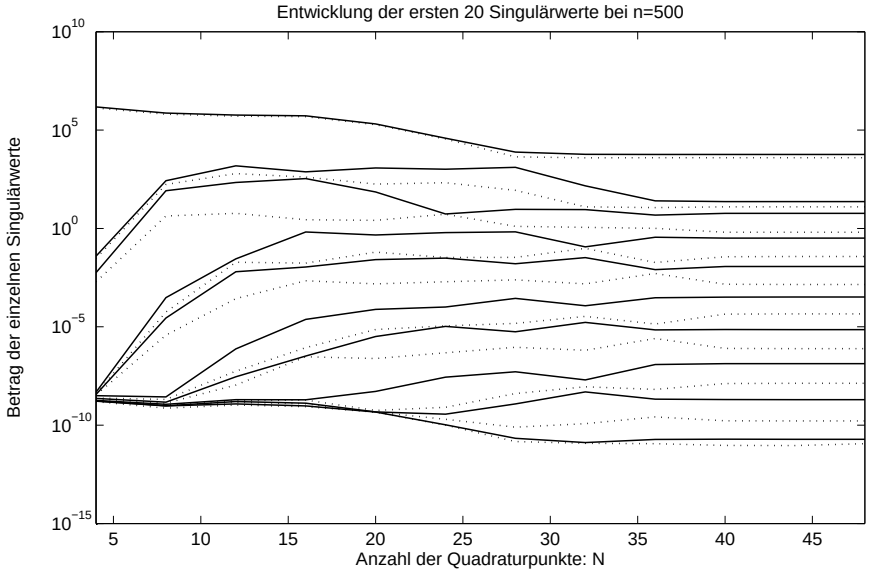


Abbildung 5.18: Entwicklung der größten Singulärwerte von K, $n = 500$

5.6 sind die größten Singulärwerte des Korrekturterms K zu den Problemen der Dimension $n = 50, 100, 500$ gegeben. Es ist jeweils ein deutlicher Abfall der Größenordnung der Singulärwerte zu erkennen. Hierbei ist ein geringer Anstieg des numerischen Rangs von K bei steigender Dimension n zu beobachten.

Die Entwicklung der größten Singulärwerte in Abhängigkeit der Anzahl an Quadraturpunkten ist für $n = 100$ bzw. $n = 500$ in den Abbildungen 5.17 und 5.18 dargestellt. Es ist zu erkennen, dass die von der Größenordnung relevantesten Singulärwerte ab 30 (für $n = 100$) bzw. 36 (für $n = 500$) Quadraturpunkten kaum noch einer Änderung unterliegen. Dies ist hauptsächlich darin begründet, dass es sich hierbei um reelle, symmetrische Probleme handelt.

5.7 Zusammenfassung

In diesem Teil der Arbeit ist ein erster Einblick zur Rang-1 Aufdatierung von Matrixfunktionen gegeben. Der Schwerpunkt dieser Arbeit wurde auf den Spezialfall der Matrixexponentialfunktion gelegt. Es wird zunächst die Aufdatierung unter Verwendung der Cauchy-Integraldarstellung von Matrixfunktionen untersucht. Als ein erstes Ergebnis stellt sich heraus, dass die Änderung der Matrixexponentialfunktion bei Rang-1 Modifikation des matrixwertigen Arguments A numerisch von niedrigem Rang ist. Es wird gezeigt, dass bei deren Berechnung durch das Cauchy-Integral die summierte Trapezregel ein geeignetes Integrationsverfahren ist. Bei einer geeigneten Wahl des Integrationswegs liefert diese einen exponentiellen Abfall des Quadraturfehlers. Unter welchen Bedingungen dies gewärleistet ist, wird durch theoretische Betrachtungen sowie anhand zahlreicher numerischer Beispiele dargestellt. Es wurde eine Möglichkeit gezeigt, wie ein geeigneter Integrationsweg durch den Wertebereich der Matrizen bestimmt werden kann. Zu dessen Berechnung wird ein effizienter Algorithmus vorgeschlagen, welcher unter anderem auf der Verwendung von Eigenwertapproximationen durch Gerschgorin-Kreise beruht. Um den Berechnungsaufwand zur Aufdatierung gering zu halten, wird des Weiteren betrachtet, wie die bei der Quadratur auftretenden geshifteten Gleichungssysteme effizient gelöst werden können.

Im zweiten Teil des Kapitels wird ein Aufdatierungsalgorithmus der Matrixexponentialfunktion dargestellt, welcher auf dem Scaling und Squaring Verfahren beruht. Dieser Algorithmus kann durchgeführt werden, ohne dass, mit Ausnahme der abschließenden Skalierung, Matrix-Matrix Multiplikationen nötig sind. Dies ermöglicht eine effiziente Berechnung der aufdatierten Matrixexponentialfunktion in $\mathcal{O}(sn^2)$ Operationen. Es wird im Zusammenhang mit dem Skalierungsparameter s gezeigt, unter welchen Bedingungen eine solche Aufdatierung gute Approximationen an die Lösung liefert. Diese Lösungsgenauigkeit wird anhand verschiedener Beispiele dargestellt. Der grundlegende Nachteil dieses Verfahrens ist, dass verschiedene Komponenten aus der Berechnung der ursprünglichen Exponentialfunktion e^A vorhanden sein müssen und nachträglich nicht mehr für das Problem e^{A+ab^T} angepasst werden können. Stellt sich hierbei im Speziellen der Skalierungsparameter s als ungeeignet heraus, ist es je nach Anwendungsfall nötig, eine explizite Neuberechnung der aufdatierten Matrixexponentialfunktion durchzuführen oder die auf dem Cauchy-Integral basierende Methode zu verwenden.

Kapitel 6

Zusammenfassung

In dieser Arbeit wurde die Rang-1 Aufdatierung von Matrizen in drei verschiedenen Themengebieten betrachtet. In diesem Zusammenhang lag der Schwerpunkt auf der Entwicklung effizienter und stabiler Verfahren mit welchen aus bereits vorhandenen Werten sowie der Rang-1 Korrektur die neue aufdatierte Lösung berechnet werden kann. Durch Verwendung dieser Algorithmen konnte der Rechenaufwand im Vergleich zur expliziten Neuberechnung in jedem der Gebiete signifikant reduziert werden. Im Einzelnen konnten folgende Ergebnisse erzielt werden.

Es wurden neue Methoden zur Rang-1 Aufdatierung der LU-Faktorisierung vorgestellt. Hierbei wurde unter anderem ein Algorithmus entwickelt, welcher die Verwendung von Spaltenpermutationen ermöglicht. Damit konnten bereits vorhandene Verfahren in stabiler Weise auf rechteckige Matrizen übertragen werden. Es wurde weiterhin ein hybrider Algorithmus dargestellt, welcher speziell an das Optimierungspaket Lrambo adaptiert wurde. Durch die Verwendung dieser Methoden ist die Berechnung der aufdatierten Zerlegungen mit einem Rechenaufwand von $\mathcal{O}(n^2)$ Operationen möglich. Dies entspricht der Reduktion um eine Größenordung im Vergleich zur expliziten Neuberechnung der Zerlegungen.

Im zweiten Teil wurde ein Verfahren zur Rang-1 Aufdatierung der Singulärwertzerlegung vorgestellt. Dieses ermöglicht die Berechnung der aufdatierten Faktorisierung in $\mathcal{O}(n^2 \log^2 n)$ Operationen. Im Fall eines Rangdefekts der betrachteten Matrix wird der benötigte Rechenaufwand weiter reduziert. Dies ist eine deutliche Ersparnis gegenüber der expliziten Neuberechnung der Zerlegung sowie auch im Vergleich zu bereits bekannten Aufdatierungsmethoden. Durch einen abschließenden Korrekturschritt kann eine sehr gute Genauigkeit der berechneten Matrizen garantiert werden. Ein Schwerpunktthema

im Rahmen dieser Aufdatierung war die effiziente Berechnung des Produkts von Cauchy-Matrizen mit Matrizen. Dieses Problem konnte durch die Verwendung von Matrixapproximationstechniken gelöst werden.

Abschließend wurde die Rang-1 Aufdatierung der Matrixexponentialfunktion betrachtet. Es wurde festgestellt, dass die Rang-1 Korrektur des matrixwertigen Arguments der Exponentialfunktion numerisch eine Niedrigrangaufdatierung des neuen Funtionswerts bedingt. Zu dessen Berechnung wurden zwei Algorithmen vorgestellt. Zum einen erfolgte die Berechnung durch die Verwendung des Cauchy-Integrals. Hierbei lag der Schwerpunkt auf der Untersuchung der Fehlerentwicklung bei der numerischen Integration. Es wurde beobachtet, dass bei der Wahl eines geeigneten Integrationswegs eine relativ hohe Genauigkeit der aufdatierten Lösung bereits mit einer geringen Anzahl an Stützstellen erzielt werden kann. Des Weiteren wurden erste Möglichkeiten dargestellt, mit welchen der relativ hohe Rechenaufwand dieser Methode reduziert werden kann. Die Grundlage des zweiten Aufdatierungsverfahrens war der Scaling und Squaring Algorithmus. Dieses ermöglicht eine effiziente Aufdatierung unter der Bedingung, dass bestimmte Werte aus vorherigen Berechnungen bekannt sind. Liegt eine geeignete Wahl des Skalierungsparameters vor, kann mit diesem Algorithmus die Lösungen im Bereich der Maschinengenauigkeit bestimmt werden.

Die theoretischen Ergebnisse, welche in den einzelnen Themengebieten erzielt wurden, konnten jeweils durch numerische Berechnungen belegt werden. Dazu wurden sämtliche hier dargestellte Algorithmen im Rahmen dieser Arbeit in C bzw. MATLAB implementiert.

Alle hier dargestellte Verfahren sind für Probleme großer, dichtbesetzter Matrizen konzipiert. Aus diesem Grund wurde nicht betrachtet, wie die hier dargestellten Algorithmen in Bezug auf Dünnbesetztheitsstrukturen weiter optimiert werden können. Da der überwiegende Teil der auftretenden Berechnungen sequentiell durchgeführt werden muss, wurde auf eine Parallelisierung der neu entwickelten Methoden verzichtet.

Literaturverzeichnis

[1] A. Aho, J. Hopcroft, und J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] A. Al-Mohy und N. Higham, *A New Scaling and Squaring Algorithm for the Matrix Exponential*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 970–989.

[3] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Advances in Design and Control, SIAM, 2005.

[4] V. Arnold, *Gewöhnliche Differentialgleichungen*, vol. 2, Springer, Berlin, 2001.

[5] G. Baker und P. Graves-Morris, *Padé approximants*, vol. 2nd Edition, Cambridge University Press, 1996.

[6] J. Bennett, *Triangular Factors of Modified Matrices*, NumerMath, 7 (1965), pp. 217–221.

[7] L. Bodrog, A. Horváth, und M. Telek, *Moment characterization of matrix exponential and Markovian arrival processes*, Annals of Operation Research, 160 (1985), pp. 51–68.

[8] H. Bowdler, R. Martin, G. Peters, und J. Wilkinson, *Solution of Real and Complex Systems of Linear Equations*, Numer. Math., 8 (1966), pp. 217–234.

[9] M. Box, D.Davies, und W. Swann, *Non-linear optimization techniques*, Mathematical and statistical techniques for industry, Imperial Chemical Industries Ltd by Oliver & Boyd, 1969.

[10] D. Braess, *Nonlinear approximation theory*, Springer Verlag, New York, 1986.

[11] D. Braess und W. Hackbusch, *Approximation of 1/x by exponential sums in [1,∞)*, IMA Journal of Numerical Analysis, 25 (2005), pp. 685–697.

[12] R. Brent und F. Luk, *The Solution of Singular value and Symmetric Eigenvalue Problems on Multiprocessor Arrays*, SIAM J. Sci. and Stat. Comp., 6 (1985), pp. 69–84.

[13] J. Bunch und C. Nielsen, *Updating the Singular Value Decomposition*, Numerische Mathematik, 31 (1978), pp. 111–129.

[14] J. Bunch, C. Nielsen, und D. Sorensen, *Rank-One Modification of the Symmetric Eigenproblem*, Numerische Mathematik, 31 (1978), pp. 31–48.

[15] A. Conn, N. Gould, und P. Toint, *Convergence of quasi-newton matrices generated by the symmetric rank one update*, Mathematical Programming, 50 (1991), pp. 177–196.

[16] C. Cowen und E. Harel, *An Effective Algorithm for Computing the Numerical Range*, tech. rep., Department of Mathematics, Purdue University, 1995.

[17] P. Deuflhard und A. Hohmann, *Numerische Mathematik I, Eine algorithmisch orientierte Einführung*, deGruyter Lehrbuch, 2002.

[18] E. Estrada und H. Hatano, *Communicability in complex networks*, Phys. Rev. E, 77 (2007).

[19] E. Estrada und D. Higham, *Network properties revealed through matrix functions*, SIAM Rev., 52 (2010), pp. 696–714.

[20] E. Estrada und J. Rodríguez-Velázquez, *Subgraph centrality in complex networks*, Phys. Rev. E, 71 (2005).

[21] A. Frommer und U. Gläßner, *Restarted GMRES for Shifted Linear Systems*, SIAM J. Sci.Comput., 19 (1998), pp. 15–26.

[22] N. Gastinel, *Inversion d'une matrice généralisant la matrice de Hilbert*, Chiffres, 3 (1960), pp. 149–152.

[23] A. Gerasoulis, *A Fast Algorithm for the Multiplication of Generalized Hilbert Matrices with Vectors*, Mathematics of Computation, 50 (1988), pp. 179–188.

[24] A. Gerasoulis, M. Grigoriadis, und L. Sun, *A fast algorithm for Trummer's problem*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 135–138.

[25] S. Gerschgorin, *Über die Abgrenzung der Eigenwerte einer Matrix*, Izv. Akad. Nauk. UdSSR Otd. Fiz.-Mat. Nauk, 7 (1931), pp. 749–754.

[26] G. Golub, *Some Modified Matrix Eigenvalue Problems*, SIAM Rev., 15 (1973), pp. 318–334.

[27] ——, *Trummer Problem*, SIGACT News, ACM Special Interest Group on Automata and Computability Theory, 17 (1985), pp. 17.2–12.

[28] G. Golub und W. Kahan, *Calculating the Singular Values and Pseudo-Inverse of a Matrix*, SIAM J. Num. Anal., (1965), pp. 205–224.

[29] G. Golub und C. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, second ed., 1989.

[30] R. Gonzales und R. Woods, *Digital Image Processing*, Addison-Wesley, 1992.

[31] L. Grasedyck und W. Hackbusch, *Construction and arithmetics of h-matrices*, Computing, 70 (2003), pp. 295–334.

[32] ——, *A Multigrid Method to Solve Large Scale Sylvester Equations*, SIAM Journal on Matrix Analysis and Applications, 29 (2007), pp. 870–894.

[33] A. Griewank, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, Frontiers in Appl. Math. SIAM, 2000.

[34] A. Griewank und M. Korzec, *Some Theory and Pseudo Routines for L-RAMBO's Linear Algebra Concerning the Handling of Inequalities*, preprint, HU Berlin, 2005.

[35] A. Griewank und A. Walther, *On Constrained Optimization by Adjoint based quasi-Newton Methods*, Optim. Methods Softw., 17 (2002), pp. 869–889.

[36] A. Griewank, A. Walther, und M. Korzec, *Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians*, Optim. Methods Softw., 22 (2007), pp. 279–295.

[37] M. Gu und S. Eisenstat, *A Stable and Fast Algorithm for Updating the Singular Value Decomposition*, research report yaleu/dcs/rr-966, Yale University, Dept. of Computer Science, 1993.

[38] ——, *A Stable and Efficient Algorithm for the Rank-One Modification of the Symmetric Eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.

[39] ——, *Downdating the Singular Value Decomposition*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 793–810.

[40] I. Gutman und A. Graovac, *Estrada index of cycles and paths*, Chemical Physics Letters, 436 (2007), pp. 294–296.

[41] W. Hackbusch, *A sparse matrix arithmetic based on h-matrices. part i: introduction to h-matrices*, Computing, 62 (1999), pp. 89–108.

[42] ——, *Entwicklungen nach Exponentialsummen*, Tech. Rep. 4, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2009.

[43] ——, *Hierarchische Matrizen, Algorithmen und Analysis*, Springer Verlag, 2009.

[44] S. Hammarling, *The singular value decomposition in multivariate statistics*, ACM Signum Newsletter, 20 (1985), pp. 2–25.

[45] P. Henrici, *Applied and Computational Complex Analysis III*, vol. III, Wiley, 1986.

[46] N. Higham, *The Scaling and Squaring Method for the Matrix Exponential Revisited*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1179–1193.

[47] ——, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[48] W. Hock und K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lectures Notes in Econom. and Math. Systems, (1987).

[49] C. Jacobi, *Über ein leichtes Verfahren Die in der Theorie der Säcularstörungen Vorkommenden Gleichungen Numerisch Aufzulösen*, Crelle's J., 30 (1846), pp. 51–94.

[50] J.Nocedal und S. Wright, *Numerical Optimization*, Springer Series in Operation Research, Springer verlag, 1999.

[51] C. Johnson, *Numerical Determination of the Field of Values of a General Complex Matrix*, SIAM J. Numer. Ana., 15 (1978), pp. 595–602.

[52] A. Kielbasinski und H. Schwetlick, *Numerische lineare Algebra*, Verlag Harri Deutsch, 1988.

[53] R. Kress, *Numerical Analysis*, Springer-Verlag, 1998.

[54] H. Kuhn und A. Tucker, *Nonlinear Programming*, Proceedings of 2nd Berkeley Symposium, (1951), pp. 481–492.

[55] F. Lewis, *Applied Optimal Control and Estimation*, Prentice Hall PTR, 1st ed., 1992.

[56] J. Lozano und F. Fernandez, *Application of singular value decomposition to the analysis of alpha-particle spectra*, Nuclear Instruments and Methods in Physics Research, A 413 (1998), pp. 233–238.

[57] A. Meister, *Numerik linearer Gleichungssysteme*, vieweg Verlag, 2005.

[58] C. Moler und C. Loan, *Nineteen Dubious Ways to Compute the Exponential of a Matrix*, SIAM Review, 20 (1978), pp. 801–836.

[59] ——, *Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later*, SIAM Review, 45 (2003), pp. 3–46.

[60] M. Moonen und B. De Moor, *SVD and Signal Processing, III: Algorithms, Architectures and Applications*, Elsevier Science, 1995.

[61] V. Y. Pan, M. A. Tabanjeh, Z. Q. Chen, S. Providence, und A. Sadikou, *Transformations of Cauchy Matrices, Trummer's Problem and a Cauchy-Like Linear Solver*, in Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel, London, UK, 1998, Springer-Verlag, pp. 274–284.

[62] Q.I.Rahman und G.Schmeisser, *Characterization of the Speed of Convergence of the Trapezoidal Rule*, Numer. Math., 57 (1990), pp. 123–138.

[63] E. Saff, *On the degree of best rational approximation to the exponential function*, J. Approximation Theory, 9 (1973), pp. 97–101.

[64] P. Stange, *Aufdatierung LU-basierter Faktorisierungen von KKT-Matrizen*, Diplomarbeit, Institut für Wissenschaftliches Rechnen, TU Dresden, 2005.

[65] ——, *On the Efficient Update of the Singular Value Decomposition*, in 79th Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM), vol. 8, Bremen, 2008, PAMM, pp. 10827–10828.

[66] ——, *On the Efficient Update of the Singular Value Decomposition*, tech. rep., TU Braunschweig, Institut Computational Mathematics, 2011.

[67] P. Stange, A. Griewank, und M. Bollhöfer, *On the efficient update of rectangular LU-factorizations subject to low rank modifications*, Electronic Transactions on Numerical Analysis, 26 (2007), pp. 161–177.

[68] G. Stewart, *Matrix Algorithms*, vol. 2, SIAM, 2001.

[69] J. Stoer, *Einführung in die Numerische Mathematik I*, Springer-Verlag, 1983.

[70] R. Thompson, *The behavior of eigenvalues and singular values under perturbations of restricted rank*, Linear Algebra and Appl., 13 (1976), pp. 69–78.

[71] L. Trefethen und M. Embree, *Spectra and Pseudospectra, The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, 2005.

[72] N. Tsao, *A Note on Implementing the Householder Transformations*, SIAM J. Num. Anal., 12 (1975), pp. 53–58.

[73] H. van Kempen, *On Quadratic Convergence of the Special Cyclic Jacobi Method*, Numer. Math., 9 (1966), pp. 19–22.

[74] J. Weideman, *Numerical Integration of Periodic Functions: A Few Examples*, The American Mathematical Monthly, 109 (2002), pp. 21–36.

[75] J. Wilkinson, *Almost Diagonal Matrices with Multiple or Close Eigenvalues*, Lin. Alg. and its Applic., 1 (1968), pp. 1–12.

[76] T. Wright, *Eigtool*, http://www.comlab.ox.ac.uk/pseudospectra/eigtool/, (2002).